UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

GESTOR DE NEGOCIOS HOSTELEROS EN ANDROID

Lorena Aguilar Briz
Tutor: Eloy Anguiano Rey

Julio de 2016

GESTOR DE NEGOCIOS HOSTELEROS EN ANDROID

Autor: Lorena Aguilar Briz
Tutor: Eloy Anguiano Rey

Departamento de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid

Julio de 2016

Lorena Aguilar Briz
Gestor de negocios hosteleros en Android

Lorena Aguilar Briz
Calle Mosquilona 59, 4^{o}A, Colmenar Viejo, Madrid

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

Dedicado a mi abuelo.
Estés donde estés, te quiero.

"Todas las piezas deben unirse sin ser forzadas. Debe recordar que los componentes que está reensamblando fueron desmontados por usted, por lo que si no puede unirlos debe existir una razón. Pero sobre todo, no use un martillo"

– Manual de mantenimiento de IBM, año 1925

Agradecimientos

Mucho tiempo ha pasado desde que emprendí el camino que me llevaría a este día. Pisé por primera vez la escuela el 9 de Septiembre de 2012 y ya me da miedo no volver a pisarla, pues al final de todo esto se ha convertido en mi segunda casa o, mas bien en la primera, y mi casa pasó a un segundo plano. Se que mis padres estarán orgullosos pero no más que dichosos por volver a verme paseando por casa. Cuatro largos años en los que la informática me ha dado tanto, y tambien me ha quitado mucho. Pero llegados al final, solo queda recoger los frutos de ese esfuerzo. Sabiendo lo que se hoy, no habria tomado una decisión diferente hace 4 años, pese a las pérdidas.

Quiero empezar este punto agradeciendoles a mis abuelos y mis padres que hoy este aquí. Mi abuelo ya no podrá verlo y lo mínimo que puedo hacer es mencionarle como merece. Ellos me han dado siempre el apoyo moral y económico que he necesitado para dedicarme enteramente a la carrera y no tener otras preocupaciones que no fuesen propias de un estudiante. También me han aguantado con los nervios y el estrés, con los gritos ante las violaciones de segmento, y nunca han dejado de apoyarme, sin apenas preguntarme para no agobiarme. A ellos en gran mayoría se lo debo todo. Yo nunca quise estudiar una carrera, no me veía capaz, decía que era para gente “lista” y se que hoy es una superación personal para mi y también para ellos.

En esta carrera también me he enamorado y ha sido del pobre que más ha tenido que aguantarme por lo que no puede faltar aquí. Le conocí en tercero de carrera, entre prácticas y más prácticas, sin saber como, sin tener tiempo, y así empezo todo. A nadie más que a él puedo agradecerle el apoyo en los peores momentos de la carrera, en las prácticas, en las asignaturas y en particular en este proyecto. Pero sería hipócrita por mi lado agradecerle el apoyo y no la paciencia, que es de lo que más se ha cargado conmigo.

El y mi familia han constituido un apoyo fundamental.

No me puedo dejar a mis amigos que pese a no verme durante meses me han guardado el sitio. Comportandose conmigo como si nunca hubiese faltado a una reunión, apoyandome incansablemente. Todas esas tardes en una terracita al sol que he tenido que rechazar, en un bar calentitos con un chocolate cuando era invierno. Me he perdido tantos momentos con ellos que solo espero poder recuperarlos ahora que empezaré a obtener los frutos de estos sacrificios.

Ahora, aunque los deje para el final no quiero que adquieran menor importancia. El último agradecimiento debo destinarlo a mis profesores y compañeros de la carrera. Aquellos profesores que se han desvivido por nuestra educación y tambien aquellos con los que nos hemos metido un poco por los pasillos. Todos nos han dado lo mejor de ellos para enseñarnos lo mejor posible. Dentro de estos profesores destacaré a mi tutor, sin el que habría sido muy dificil este proyecto, ha supuesto un gran apoyo. Y ahora mis compañeros, compañeros de llanto, de alegrías, de revisiones, de suspensos, de competencia. Todos con un mismo fin hemos luchado en esta carrera y muchos, por desgracia, sin comprender que nosotros mismos eramos nuestro mejor apoyo. Yo he tenido la suerte de contar con compañeros excelentes que me han explicado cosas, me han ayudado en prácticas, etc. Y espero poder haberles ayudado del mismo modo, pues al final creo que como informáticos, el trabajo en equipo es fundamental.

Resumen

En la actualidad existen todo tipo de herramientas en el mercado que de una forma u otra ayudan a los negocios de hostelería a gestionar su personal, productos, proveedores, etc. Aun así, gran parte de estas herramientas mejoran la gestión de los establecimientos orientándolas a sus empleados y no a sus clientes. Existen herramientas más orientadas a los clientes pero por diversos motivos los establecimientos no hacen uso de ellas.

En relación con las necesidades que puedan tener los clientes de estos negocios se ha desarrollado ***The Waiter***, una herramienta diseñada en Android para dispositivos móviles que permite integrar las tecnologías en los negocios de hostelería, tecnologías que agilizan las esperas de los clientes independizando sus peticiones de aquel personal destinado a atenderlas. Además permite una abierta comunicación entre todos los usuarios conectados a la red local del establecimiento. Esta herramienta es intuitiva y fácil de usar: no genera dudas en los usuarios.

El sistema se ha implementado en Android para dispositivos móviles por ser uno de los medios más utilizados. Entre los motivos más importantes por los que se ha elegido esta tecnología destacan los bajos costes y la facilidad de ser implantada en diferentes negocios. ***The Waiter*** es un sistema para agilizar la gestión de los negocios destinados a la hostelería y más concretamente al área de alimentación. Permite que los clientes puedan realizar todo tipo de peticiones a los camareros a través de sus dispositivos móviles y estos, a su vez, podrán visualizarlas en los suyos. También proporciona un medio de comunicación entre usuarios y empleados conectados a la red local de un establecimiento, a través de un chat público y un servicio de mensajería instantánea entre dos usuarios. Además, ***The Waiter*** permite a los clientes realizar distintas valoraciones acerca del servicio recibido por parte del negocio y el administrador del mismo tendrá acceso a esta información para valorar la calidad de la atención prestada. Todas estas funcionalidades se implementan con un protocolo de comunicación diseñado especialmente para este proyecto, esto requiere únicamente la configuración de una red local en el establecimiento con acceso a internet para que los clientes puedan descargar la información necesaria del negocio desde la base de datos. Por ello la implantación de esta herramienta conlleva unos costes muy bajos, lo que facilita su integración en el mercado.

The Waiter se presenta como una solución real y original a los problemas cotidianos de los establecimientos de hostelería, así como un medio al alcance de cualquier individuo para mejorar y flexibilizar las pérdidas de tiempo innecesarias en su día a día.

Palabras clave

Aplicación móvil, Android, Java, Código QR, Smartphone, Hostelería, Mensajería, Chat, Multicast.

Abstract

Nowadays there are a lot of tools in the market that, in many ways way, help the hostelry businesses managing their staff, products, providers and more. Even so, most of these tools improve the establishment's employee management rather than the client side. There are tools more oriented to the clients but, because of several reasons, the establishments do not make use of them.

The Waiter has been developed to deal with the needs that the clients of this business may have. ***The Waiter*** is an Android tool designed for smartphones that allows integration of the technologies in the hostelry business. These technologies shorten the client's waiting time by creating a petition that is independent of the employer destined to carry out the assigned task. Furthermore, it allows an open communication between all the users connected to the establishment's local network. This tool is intuitive and user-friendly.

The Waiter has been developed for Android since it is the most used platform. The most important reasons why this platform has been chosen, are the low costs and the ease of insertion in different businesses. ***The Waiter*** is a method to streamline the management of the hostelry businesses, more precisely the food business. It allows the clients to make all kind of requests to the waiters through their mobile phones. The waiters will also be able to visualize these requests in their own devices. It creates a communication channel between users and employees connected to an establishment's local network through a public chat and an instant messaging service between two users. Besides, ***The Waiter*** allows the clients to rate the received service, hence, the establishment's manager will be able to access this information to assess the quality of his customer service. All of these functionalities are implemented with a communication protocol specially designed for this project. It only requires a local network configuration in the establishment with internet access to allow the clients to download the necessary information from the database. Due to all these reasons, the tool's installation has a very low cost, which makes its market insertion easier.

The Waiter is presented as a real and original solution to the everyday problems of the hostelry establishments, as well as an accessible tool to everyone that will improve unnecessary time lost in an establishments day to day routine.

Keywords

Mobile aplication, Android, Java, QR code, Smartphone, Hostelry, Messaging, Chat, Multicast.

Índice

Listas

Lista de códigos

Lista de figuras

Lista de tablas

1

Introducción

1.1. Marco del proyecto

La situación actual de los negocios dedicados a la hostelería se centra en una atención personalizada mediante recursos humanos, el uso de tecnologías por parte de estos establecimientos es mínimo.

La atención personalizada que proporcionan estos negocios de hostelería tiene gran acogida entre los clientes pero a veces el número de personal frente al número de solicitudes resulta insuficiente para atender todas estas de manera rápida y eficiente.

The Waiter se centra en solventar los problemas que surgen en los negocios más tradicionales puesto que son, por lo general, los que menos tecnologías utilizan centrándose más en la atención personalizada. Estos negocios por lo general siguen un sistema simple para atender a los clientes: en la primera atención se recoge la solicitud del cliente y en la segunda se atiende esta solicitud de la forma correspondiente. Las herramientas tecnológicas que utilizan abarcan la gestión de los pedidos y los cobros.

Este sistema resulta lento cuando el número de clientes supera en demasía al número de empleados. El problema nace de la necesidad de los empleados de acudir dos veces a la mesa para realizar un único servicio, lo que claramente provoca efectos muy negativos en el tiempo que transcurre entre la petición y la respuesta.

1.2. Motivación

Escoger un tema sobre el que realizar un trabajo de esta magnitud no ha sido una tarea fácil. Pese a las múltiples ideas que han ido surgiendo durante años mientras se acercaba este momento, a la hora de enfrentarse a ello, nada parecía suficiente. El proyecto que se ha decidido llevar a cabo es una aplicación en Android que persigue el mayor rendimiento de los usuarios en las actividades cotidianas de mayor frecuencia y menor relevancia, optimizando así el tiempo que requieren determinadas tareas, básicamente, ahorrando a los usuarios esperas innecesarias.

En lo que se refiere a la aplicación, se podría pensar que es “otra más”, sin embargo, hay ciertos detalles que permiten distinguir a este árbol en medio del bosque. Estos detalles fundamentalmente se deben al enfoque de la aplicación, como se explicará a lo largo de este documento.

En los últimos años el número de tecnologías dedicadas a mejorar la gestión de los negocios de hostelería ha aumentado considerablemente. Gracias en gran parte al crecimiento del número de negocios “de comida rápida” que son conocidos así por la agilidad en el servicio, debido en su mayoría al uso de herramientas electrónicas. Siendo así, muchas personas se podrían plantear que este proyecto trata de reinventar la rueda pero ababaremos demostrando que no es así. La aplicación que se presenta es una herramienta que permite evitar determinadas esperas en los negocios dedicados a la hostelería que tanto se frecuentan a diario, como por ejemplo aquellas

que se producen cuando los clientes esperan a ser atendidos para realizar una solicitud,

El resultado de este planteamiento ha sido el desarrollo de una herramienta que pueda llevarse a la práctica obteniendo resultados que denoten la mejora del rendimiento esperado, lo que supone una reducción de los tiempos de servicio que se ofrecen en la actualidad, haciendo a la herramienta lo más sencilla, accesible y atractiva posible. Además de esto, la herramienta ***The Waiter*** presenta costes muy bajos para su implantación que pueden potenciar su uso en detrimento de otros medios mucho más costosos actualmente en uso.

1.3. Objetivos

El objetivo principal de este proyecto es, por tanto, diseñar e implementar una herramienta para los negocios de hostelería que gestione de una manera eficiente las peticiones de los clientes e implante un sistema de comunicación entre trabajadores y clientes conectados en la red privada del establecimiento.

Todos estos objetivos se concentran en ***The Waiter***. ***The Waiter*** es la herramienta de "Gestión de negocios de hostelería" desarrollada para este trabajo de fin de grado. La aplicación es capaz de gestionar las peticiones y valoraciones de los usuarios concentrando funcionalidades diferentes en función del rol del usuario que tenga acceso a ella, está preparada con un rol que permite acceder a la configuración de las preferencias de cada negocio, un rol de camarero que permite acceder a la lista peticiones de los clientes y un rol de cliente que permitirá a los usuarios realizar diferentes peticiones así como establecer una comunicación interna con otros usuarios del local. Todo esto con una interfaz gráfica desarrollada en Android que permite a los usuarios utilizar las funcionalidades que ofrece la aplicación de forma intuitiva.

1.4. Estructura del documento

El en capítulo 2 se hace un análisis del estado del arte, en el que se detalla la situación actual de los negocios de hostelería, las herramientas que existen en la actualidad similares a la que se presenta en estre proyecto y una breve introducción sobre Android, la base fundamental de este trabajo.

En el capítulo 3 se detallan los objetivos y funcionalidades de este proyecto.

En el capítulo 4 se define la aplicación que se ha diseñado. Para ello se detalla la metodología que se ha utilizado en el desarrollo así como todas las herramientas que han formado parte del mismo.

Los capítulos 5, 6 y 7 componen la mayor parte del contenido de este documento. En el capítulo 5 se hace un detalle del análisis realizado incluyendo el catálogo completo de requisitos y los casos de uso para los usuarios de la aplicación. En el capítulo 6 se desarrollan el diseño y la implementación. Para ello se detallan la arquitectura utilizada y sus componentes, el diseño de la base de datos, la navegación entre pantallas, etc. En el capítulo 7 se hace un análisis de las pruebas que se han realizado sobre la aplicación.

En el capítulo 8 se evalúa el resultado final por parte del equipo de desarrollo y los usuarios escogidos y se identificarán los beneficios que aporta la aplicación.

En el capítulo 9, como punto final, se detallan las conclusiones obtenidas del proyecto así como un breve planteamiento sobre el trabajo futuro.

2

Estado del arte

2.1. Negocios de hostelería

Hostelería es el nombre genérico que reciben las actividades económicas consistentes en la prestación de servicios ligados al alojamiento y la alimentación, usualmente prácticas ligadas al turismo.

Se han distinguido negocios de 2 tipos según el número de tecnologías que utilizan en sus servicios.

2.1.1. Tradicionales

Los negocios tradicionales se valen de muy pocas tecnologías para mejorar los servicios que ofrecen, en su lugar se utilizan muchos recursos humanos. ***The Waiter*** está orientada a aquellos negocios de hostelería que ofertan servicios de alimentación, como pueden ser los restaurantes o los bares. La mayoría de estos negocios incorporan una tecnología denominada TPV (Terminal Punto de Venta) que agiliza las tareas de atención a los clientes mediante la acumulación de los pedidos, la elaboración de facturas rápidas y el almacenaje de datos relevantes para el establecimiento, por ejemplo los productos más solicitados.

2.1.2. De comida rápida

Este tipo de negocios incorpora cada vez un mayor número de tecnologías pero se orientan en que los clientes hagan su propio pedido ahorrandose así recursos humanos pero dificultando en muchas ocasiones la acción de los usuarios. Benefician la agilidad del negocio pero ponen en duda la comodidad de los clientes. Es por ello que aunque muchos negocios han incorporado los auto-servicios, siguen manteniendo el servicio tradicional.

The Waiter en la actualidad está más enfocado a los negocios tradicionales puesto que por lo general en los establecimientos donde se ofrece un “servicio rápido” no existe atención personalizada en las mesas.

2.2. Aplicaciones de gestión hostelera y sus funciones típicas

En la actualidad existe una amplia gama de aplicaciones que comparten algunos de los objetivos de este proyecto. Se citan las que se han considerado más reelevantes y cuya funcionalidad tiene más similitudes con la funcionalidad de ***The Waiter***.

The Waiter es una herramienta creada con una intención enfocada al cliente. Pese a esto, se han implementado funcionalidades que facilitan la gestión del negocio y se ha propuesto trabajo futuro para completar el alcance de la aplicación. Por ello las herramientas que se citan a continuación se dividen en dos enfoques según estén orientadas a los clientes o al negocio.

2.2.1. Destinadas al cliente

Dual Link B&R Comandero

Esta aplicación permite que los camareros de un restaurante, bar, pub o cualquier otro negocio de hostelería tengan una herramienta eficaz para realizar los pedidos así como sincronizarlos automáticamente con la aplicación de TPV, agilizando de este modo su trabajo y permitiéndoles mandar el pedido de forma seleccionada, parcial y completa.

Con Dual Link B&R Comandero los camareros tienen la oportunidad de solicitar los pedidos de los clientes a la cocina y a la barra, además, pueden seleccionar si la comida que se va a servir ha de mantenerse en frío o en caliente. También puede recibir y marcar los pedidos realizados por los clientes que estén usando la aplicación “Carta” de la Suite de Ocio y Restauración y efectuar el pago directamente desde el propio terminal (con tarjeta o móvil).

iD Bares

Esta aplicación sirve para interactuar con los locales de ocio iD. A través de ella, el usuario puede pedir comandas o bebidas desde la mesa utilizando un iPad incluido en la misma o desde su propio smartphone.

Desde la aplicación, el cliente, además, puede realizar reservas, consultar el menú, seleccionar cada ingrediente de sus consumiciones, conocer el gasto acumulado, individualizar los pagos y obtener descuentos en función de su participación en el local. Además dispone de una sección de juegos y retos.

MyOrder

Desarrollada por la empresa Atecresa en Canarias, con ella los usuarios disponen de un sistema para evitar incomodas y largas esperas en los bares y restaurantes, ya que con esta aplicación se puede realizar un pedido a través de códigos QR (Quick Response).

Gracias a la aplicación “MyOrder” se puede realizar el pedido desde el móvil o Smartphone, sin tener que esperar a que el camarero tome nota.

qlikBar

Esta aplicación es capaz de conocer el local en el que se encuentra el usuario y darle la posibilidad de consultar el menú, interactuar con el resto de clientes, dejar comentarios y opiniones y realizar pedidos. Ello permite a los clientes que el tiempo que pasen en el bar o restaurante sea más social y divertido, además de que la aplicación es también una guía de locales para ayudar a descubrir sitios nuevos y un juego en el que los usuarios acumulan puntos por visitas y consumos en los locales. Los usuarios de smartphone se puede descargar de forma gratuita la aplicación.

2.2.2. Destinadas al negocio

GlopDroid

La app es un sistema de toma de comandas muy funcional e intuitivo, ya que aprovecha la calidad gráfica de smartphones y tablets para ofrecer una forma de trabajar más cómoda y ágil en restaurantes, cafeterías y otros locales de hostelería. Permite a los camareros tomar nota de las comandas en dispositivos móviles en lugar de en papel.

Puede gestionar comentarios a cocina, enviar mensajes con sugerencias, seleccionar los platos para un menú o realizar traspasos entre mesas, entre otras muchas opciones. Además controla los salones colocando mesas o cobrando las mismas por pago fraccionado si así lo desean los clientes y gestiona tantas impresoras de cocina como sean necesarias. Equipado tan sólo con una red wifi y la aplicación de Glop para hostelería, proporciona una solución de cara a la atención al cliente.

Dual Link B&R TPV

Implementa las funcionalidades más importantes de un TPV pero ofreciendo una interfaz sencilla y usable que puede ser incorporada en cualquier dispositivo con sistema iOS.

Esta aplicación permite a los usuarios gestionar los pedidos por secciones: localizados y no localizados. También es capaz, por ejemplo, de asignar puntos VIP para los clientes más importantes.

Los pedidos localizados se administran gracias a un plano personalizable de las mesas, con el cual se pueden juntar mesas o distribuirlas a semejanza del local. Los pedidos no localizados conciernen a los taburetes y puestos en la barra, cuyos clientes estarán atendidos gracias a esta aplicación.

Philomarket

Es una app gratuita para hacer los pedidos a proveedores desde cualquier móvil o tablet. También se puede acceder a la web desde cualquier dispositivo.

QR Question

Con QR Question los hosteleros tienen acceso de forma instantánea a los comentarios y valoraciones de sus clientes. Los comensales, por su parte, acceden a una breve encuesta a través de una URL o un código QR, y desde el mismo restaurante valoran el servicio.

2.3. Android

Android [1–4], entre sus muchas definiciones, es un sistema operativo para dispositivos móviles basado en Linux.

El SDK (Software Development Kit) de Android [5] proporciona las herramientas necesarias para desarrollar aplicaciones Android utilizando el lenguaje de programación Java [6–8]. Android fue desarrollado originalmente por una startup del mismo nombre que fue comprada por Google en 2005. Cualquiera puede empezar a utilizar Android descargando su código completo. Además, los fabricantes pueden añadir sus extensiones propietarias, lo cual ha hecho crecer considerablemente la aceptacion de la plataforma.

La historia de Android comenzó con la publicación de Android beta en Noviembre de 2007. La primera versión comercial apareció en febrero de 2009. Desde abril de 2009, las versiones reciben un nombre inspirado en un postre o un dulce, en orden alfabético (tabla A.1).

La versión más reciente de Android es 6.0 Marshmallow, que hace uso de la API 23 de Android. Sin embargo no es la versión más usada actualmente, como se puede comprobar en la figura A.1.

2.4. Opiniones y conclusiones

Las aplicaciones citadas anteriormente son solo unas pocas de las que existen actualmente en el mercado, las más similares que se han encontrado al proyecto que se presenta.

Tras una larga investigación sobre la aplicación de estas herramientas se ha advertido que pese a sus múltiples ventajas no han tenido apenas éxito y es una idea muy poco explotada por los negocios de hostelería.

En su mayoría, esto se debe a la escasa aceptación por parte de los negocios de introducir nuevas herramientas, bien por los gastos económicos o bien por un temor frecuente a las nuevas tecnologías. Por ello, estas aplicaciones, que en cierto modo pretenden cambiar todo el modelo de gestión del negocio y aunque resulten una mejora, se enfrentan al rechazo frente a métodos más tradicionales.

Es por ello que la herramienta ***The Waiter*** requiere una inversión inicial prácticamente nula, el camarero podría usar su propio smartphone y los clientes utilizan sus propios dispositivos también. En cuanto a los temores por las nuevas tecnologías, se ha hecho un esfuerzo considerable por hacer una interfaz simple y usable que permita que tanto la parte del negocio como los clientes puedan utilizarla de una forma intuitiva.

3

Objetivos y funcionalidades

3.1. Introducción

Como se ha especificado en el capítulo 1, el propósito de este Trabajo de Fin de Grado es definir, implementar y presentar un sistema software de gestión hostelera y comunicación entre dicho negocio y sus clientes. Para cumplir estos objetivos se detallan las funcionalidades que dan forma a ***The Waiter***.

3.2. Objetivos específicos y funcionalidades

3.2.1. Funcionalidades

En este apartado se definen las funciones básicas del proyecto. El propósito de estas funcionalidades es dar solución a las necesidades que originan el poryecto y cubrir los objetivos planteados para el mismo.

- **Envío de peticiones al camarero.**
    - **Limpieza en mesa.** Se solicitará que el camarero acuda para limpiar el espacio de los clientes.
    - **Atención.** Se solicitará que el camarero acuda a la mesa para realizar una petición personalizada.
    - **Cuenta.** El cliente solicitará la cuenta.
    - **Pago electrónico.** El cliente solicitará al camarero que acuda a la mesa con los medios necesarios para llevar a cabo un pago con tarjeta.
    - **Pedido.** Tras ir añadiendo productos al carrito el usuario podrá solicitar el pedido completo al camarero.
    - **Pedido personalizado.** En este pedido se enviará al camarero un mensaje personalizado por si fuese necesaria la modificación de alguno de los productos de la carta o bien este no estubiese incluido.
    - **Hoja de reclamaciones.** Los clientes podrán solicitar la hoja de reclamaciones, servicio que debe estar disponible en todos los establecimientos.
- **Gestión del carrito [9–11].** En el carrito el cliente podrá ver un listado clasificado de los productos que pertenecen a la carta del negocio. En estos listados podrá ir añadiendo y eliminando productos al pedido.

- **Gestión del pedido.** En el pedido se añadiran aquellos elementos que se han ido añadiendo desde la funcionalidad de carrito. Cuando se muestre el listado de pedido tambien se permite que el usuario añada o elimine elementos y finalmente podrá enviar este pedido al camarero.
- **Gestión del perfil de usuario.** Los usuarios podrán modificar sus datos personales que serán utilizados como públicos en ciertas funcionalidades de la aplicación como el chat.
- **Envío de valoraciones y sugerencias.** Los clientes podrán valorar y hacer sugerencias que ayuden al negocio a mejorar sus servicios.
- **Escaneo de códigos QR.** La aplicación necesita conocer datos sobre el negocio así como las notificaciones que recibe un camarero necesitan incluir información sobre la localización del cliente, para ello, deben escanear un código QR situado, por ejemplo, en la mesa en la que se sienten. Los camareros deberán escanear otro parecido proporcionado por el administrador.
- **Comunicación en red local [12, 13].** Los usuarios conectados a la red local del establecimiento podrán iniciar con los otros usuarios conectados un chat público o una mensajería instantanea con cualquiera de los clientes que estén utilizando este servicio.
- **Realización del test de alcoholemia [14, 15].** Los clientes podrán rellenar un test de alcoholemia que ofrece la aplicación para aproximar la tasa de alcoholemia y poder garantizar en la medida de lo posible la seguridad de los empleados.

Solo se han detallado aquellas funcionalidades que serán comunes para todos los usuarios de la aplicación.

3.2.2. Objetivos

Las funcionalidades citadas anteriormente tienen como propósito cumplir los objetivos que se detallan en este apartado. Los objetivos de la aplicación son muchos pero pueden agruparse en los objetivos genéricos que se citan a continuación:

- **Minimizar el tiempo de espera de los clientes.** Esto se consigue haciendo que los clientes puedan realizar peticiones desde el teléfono móvil sin esperar a que un empleado del establecimiento les atienda para escuchar su solicitud y poder atenderla.
- **Comunicación de los usuarios.** El protocolo de comunicación interno que permite el envío y recepción de peticiones también se utiliza para ofrecerle a los clientes un chat público para todos aquellos que estén conectados a la red local del establecimiento. A parte de esto, ***The Waiter*** ofrece la posibilidad de iniciar hilos de comunicación privada uno-a-uno entre clientes que estén utilizando el servicio.
- **Agilizar las tareas del camarero.** El camarero no tendrá que ir mesa por mesa anotando las peticiones sino que las recibirá en un dispositivo móvil de su eleccion. Para atender estas peticiones ***The Waiter*** ofrece un sistema de prioridades, en posteriores capítulos se explicará más en detalle.
- **Reducción del tiempo de navegación mediante el desarrollo de una interfaz sencilla y usable.** Este es uno de los objetivos más importantes. Puesto que no se aporta manual para los usuarios, solo una breve comunicación con aquellos negocios que quieran implantar este sistema en sus establecimientos, se ha perseguido que la interfaz sea lo más intuitiva posible para que los usuarios conozcan sus posibilidades en todo momento y puedan hacer una navegación fácil entre las pantallas de la aplicación.

4

Definición del proyecto

4.1. Metodología

En este apartado se definirán con detalle, y en orden de ejecución, cada una de las etapas que se han seguido durante el desarrollo del proyecto. Las fases del proyecto coinciden con las que normalmente se utilizan en la mayoría de proyectos software:

1.– **Análisis.** En esta fase la tarea principal es la captura de requisitos. Estos deben definir con exactitud el funcionamiento de la aplicación. También incluye requisitos no funcionales.

2.– **Diseño.** Esta etapa esta dedicada a realizar el diseño de la aplicación. Esto incluye el diseño de pantallas, navegación, base de datos, protocolo de comunicación y otros elementos menos relevantes para la aplicación. También se ha hecho un breve diseño de la estructura de clases puesto que en su base la aplicación se ha desarrollado en Java. El diseño de la interfaz de usuario se incluye también en esta etapa.

3.– **Implementación.** Esta etapa es, con diferencia, la más larga de las que componen el proyecto. En ella se codifican los diseños que se han definido en la fase anterior. También se realizan en esta etapa las pruebas correspondientes a la implementación de cada tarea aunque el modelo de ciclo de vida utilizado nos ha permitido volver a esta fase para realizar las modificaciones que se hayan considerado oportunas. Todas las fases pueden volver a ser abiertas de nuevo para aplicar modificaciones pero esta es la fase en la que más nos ha interesado y por la que hemos elegido el modelo de ciclo de vida del software que se comentará al final de esta sección.

4.– **Pruebas.** En la etapa anterior se han realizado muchas de las pruebas previstas puesto que nos permitia cerrar la fase de implementación lo mas completa posible. En esta etapa se han completado las pruebas restantes y se han revisado las que ya se habian realizado para asegurar la funcionalidad. También se ha aporvechado esta fase para documentar las pruebas realizadas y los módulos que por su completo y correcto funcionamiento se han considerado cerrados.

5.– **Documentación.** En las etapas anteriores se ha ido realizando un gran trabajo de documentación pero es en esta fase cuando hemos reunificado todo para dar lugar a la presente documentación. En ella se recogen los aspectos más importantes del proyecto y se crea una relación entre los elementos de cada una de las fases. Finalmente el documento recoge la evaluación, extensibilidad, trabajo futuro y conclusiones obtenidas del proyecto.

6.– **Mantenimiento.** Esta fase es la más larga del proyecto. Durante ella se mejora, actualiza y modifica aquello que se considere necesario en la aplicación. En el caso de este proyecto no se ha trabajado sobre esta etapa aunque resultará necesaria para el trabajo futuro y por ello se

detallarán algunas tareas de mantenimiento que se han previsto.

En el diagrama de Gantt del apartado 4.2.2 pueden observarse estas etapas divididas en tareas. También se puede observar la distribución temporal de estas tareas así como las relaciones que existen entre ellas, las cuales permiten solaparlas en las etapas de desarrollo.

Cómo ya hemos mencionado ligeramente en la fase de implementación, se ha seguido un modelo de ciclo de vida en cascada [16, Cap. 1]. Al constar unicamente de un recurso humano se ha decidido utilizar una variante que permite solapar el desarrollo de varias tareas pudiendo volver después a abrir cualquiera de las fases cuando se necesiten realizar modificaciones sobre ella. Esta variante es una variante iterativa (figura 4.1) y con ella hemos conseguimos evitar limitaciones que se encontraron en el modelo clásico.

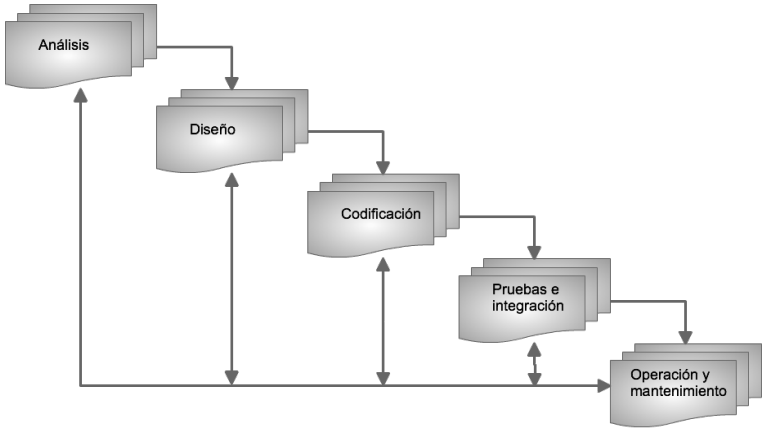


Figura 4.1: Ciclo de vida en cascada con realimentación.

4.2. Planificación orientativa

La planificación [17, Cap. 10] que se aporta a continuación es orientativa puesto que se realizó en la fase de planificación del proyecto y aunque se ha hecho todo lo posible por seguirla, en ocasiones ha sufrido alguna desviación.

Los cambios, planteados en el diagrama de Gantt, han surgido tras las revisiones y es por ello que no se plantearon en la planificación inicial sino que se incorporaron cuando fueron necesarios y así quedan especificados en el diagrama.

4.2.1. Puntos de revisión

Puesto que el equipo de desarrollo estaba compuesto únicamente por un recurso humano, las revisiones que vale la pena mencionar son las que se han realizado en conjunto con el tutor del proyecto.

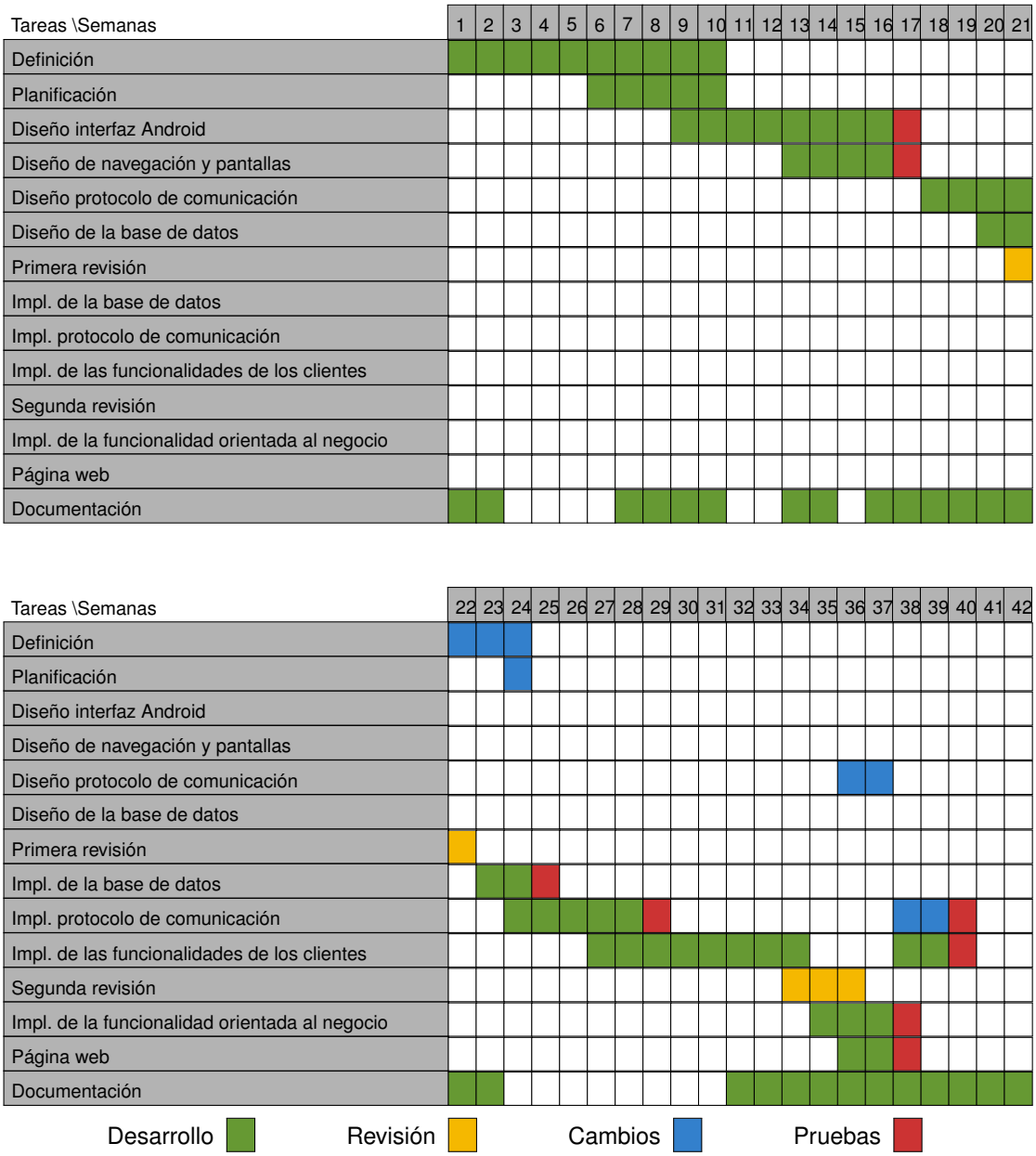
- **Semanas 21-22.** En este punto de revisión se comprueban las definiciones y los diseños para dar paso a la fase de implementación de los diferentes componentes de la aplicación.
  Tras esta revisión fue necesario incluir modificaciones en la definición y la planificación del proyecto.
- **Semanas 34-36.** En esta segunda y última revisión el objetivo es comprobar el correcto funcionamiento de los elementos que componen la aplicación. Por claridad en el diagrama se hace unicamente referencia a tareas genéricas, las cuales engloban multitud de subtareas.

Después de esta revisión se han llevado a cabo modificaciones sobre las implementaciones del protocolo de comunicación y la funcionalidad completa de los usuarios.

4.2.2. Diagrama de Gantt

El diagrama [17, Cap. 8] se ha estructurado en dos partes que, además de facilitar la lectura, permiten distiguir dos fases claves en el proyecto:

- Análisis, planificación y diseño.
- Implementación.



4.3. Herramientas utilizadas

Para la implementación de los elementos que forman la aplicación se han utilizado entornos de trabajo, herramientas para la gestión de versiones, simuladores y lenguajes de programación. Estas herramientas se detallan a continuación:

4.3.1. Plataformas

Android Studio

Android Studio [18–24] es el entorno de desarrollo integrado que se ha escogido en este proyecto para desarrollar la aplicación. La elección de esta herramienta frente a otras se debe a que está desarrollada en su base para programar en Android a diferencia de otras que permiten múltiples lenguajes. Puesto que no se tenían conocimientos previos sobre Android, se apostó por una plataforma desarrollada con este fin específico.

Genymotion

Genymotion [25] es un completo emulador para aplicaciones Android. Ha permitido ejecutar la aplicación hasta en 5 dispositivos sin alterar el rendimiento de la misma. Esto ha resultado muy útil para simular situaciones lo más cercanas posibles a la realidad de los entornos en los que se utilizará la aplicación. También ha permitido comprobar compatibilidades de interfaz y versiones Android con 7 tipos diferentes de dispositivos y 5 versiones.

Bitbucket

Es un sistema de alojamiento basado en web para aquellos proyectos que utilizan el sistema de control de revisiones Mercurial y Git [26].

En el caso se este proyecto se ha utilizado el sistema de revisiones Git. Esto ha permitido controlar y versionar todos los elementos del proyecto. Ha resultado muy útil para las revisiones y las modificaciones puesto que se ha tenido constancia de todos aquellos instantes en los que se ha subido una versión del proyecto con la correspondiente descripción. Esto ha permitido detectar rápidamente los módulos con errores, los que requerían modificar o los que se han expuesto en las correspondientes revisiones.

4.3.2. Frameworks

Android

Pese a la dificultad que supone clasificar esta herramienta se ha optado por incluirla como un marco de trabajo puesto que no es un lenguaje, sino que utiliza otros lenguajes como Java y XML, y aunque es un sistema operativo, no hemos usado el sistema como herramienta para este proyecto sino como plataforma para desplegar la aplicación.

La razón por la cual se escoge este framework en lugar de iOS o WindowsPhone es meramente práctica, en el sentido de que Android, como sistema operativo, es el más extendido [27] entre los Smartphones alcanzando un 87.8 % de la cuota de mercado en Enero de 2016 [28] (figura 4.2).

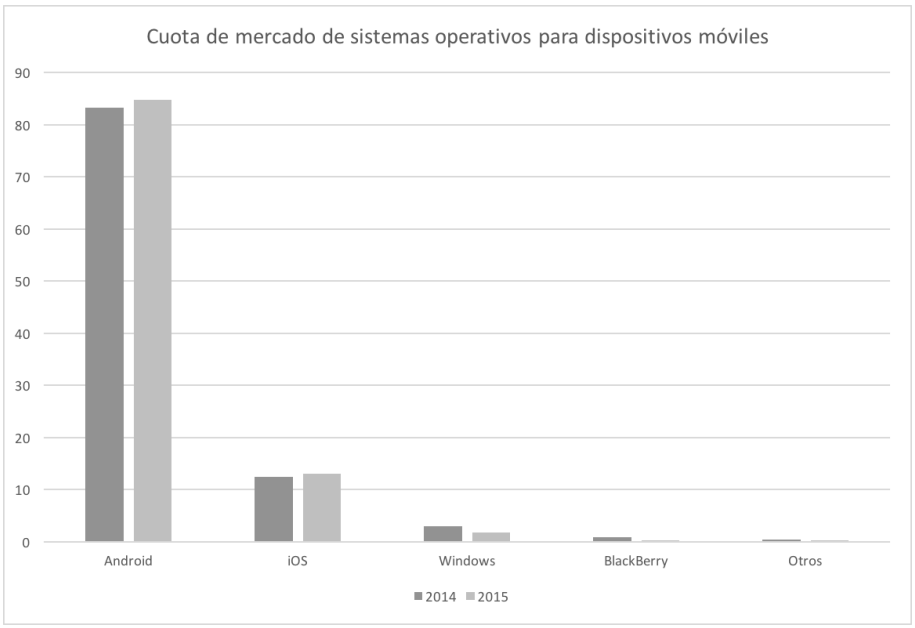


Figura 4.2: Cuota de mercado de sistemas operativos para dispositivos móviles en los últimos años.

El motivo por el que se escoge una herramienta orientada a los dispositivos móviles o tabletas es la expansión de estos dispositivos en la actualidad (tabla 4.1) [29].

País	N^o de teléfonos móviles	Población	% de la población	Última actualización
Montenegro	1,294,167	672,180	192.5	Diciembre de 2013
Hong Kong	13,264,896	7,008,900	187.9	Noviembre de 2010
Arabia Saudita	46,000,000	27,137,000	169.5	Junio de 2010
Lituania	4,960,000	3,341,966	148.4	Febrero de 2010
Estonia	1,982,000	1,340,602	147.8	Abril de 2009
Uruguay	5,000,000	3,395,000	147.2	Septiembre de 2013
Guatemala	22,700,000	15,806,675	143.6	Octubre de 2012
Bulgaria	10,655,000	7,600,000	140.2	2008
Portugal	14,500,000	10,632,000	137.0	2008
Alemania	120,000,000	81,882,342	130.1	2009

Tabla 4.1: En esta tabla se muestra el número de dispositivos móviles por paises en comparación con el número de habitantes. Únicamente se muestran los 10 que presentan mayores cifras.

4.3.3. Lenguajes de programación

SQL

SQL (Structured Query Language) [30, 31] Se trata de un lenguaje de programación declarativo de bases de datos relacionales cuyas siglas significan lenguaje de consultas estructurado y es uno de los más famosos en el mundo para la programación y el manejo de dichas bases de datos. En este trabajo de fin de grado se ha usado concretamente PostgreSQL, un sistema de gestión de bases de datos libre que utiliza este lenguaje.

HTML

HTML (HyperText Markup Language) [32, 33] es un lenguaje usado para la programación de páginas web, siendo el encargado de dar estructura y contenido a las mismas. Usa estructuras predefinidas como tablas, divisores, cabeceras, y en sus versiones más recientes, hasta sprints para animaciones sustituyendo al obsoleto Flash. Se puede combinar con otros lenguajes como Javascript, PHP y CSS para darle más estilo y funcionalidad a la página. En este caso se ha hecho uso de la versión 5 de HTML.

Esta herramienta se utiliza para elaborar páginas web.

Javascript

JavaScript [34, 35] es un lenguaje de programación orientado a objetos de scripting desarrollado por Oracle y el más usado en todo el mundo en 2015 según un informe de RedMonk (http://redmonk.com/sogrady/2015/07/01/language-rankings-6-15/). Se usa para el desarrollo estructurado tradicional o más concreta y exactamente para el desarrollo de páginas web, siendo ejecutado siempre en el lado del cliente para aprovechar la capacidad de procesamiento de éste y no sobrecargar el rendimiento del servidor. En este proyecto ha sido utilizado para validar los datos del formulario de registro para los usuarios de la aplicación.

PHP

PHP (Pre Hypertext Processor) [36, 37] se trata de un lenguaje de programación estructurado y de código abierto que tradicionalmente se utiliza para dotar de una funcionalidad mayor a las páginas web, siendo un apoyo y complemento de HTML y pudiéndose incrustar en este último.

CSS

CSS (Cascading Style Sheets) [38, 39] es un tipo de fichero XML (eXtended Markup Languaje) orientado a dar estilo a una determinada página web. Sus siglas en español significan: Hoja de estilo en cascada, se invoca desde un fichero HTML y dota del estilo seleccionado por el programador a cada elemento especificado. Es un formato muy flexible y amplio que permite modificar completamente la apariencia de una página web al gusto del usuario.

5

Análisis

5.1. Introducción

En este capítulo se presenta en detalle la fase de análisis del proyecto [16, Cap. 4, Cap. 5]. Para ello se exponen los casos de uso y el catálogo de requisitos. El catálogo de requisitos se elabora como una guía a seguir para completar la funcionalidad que se espera del proyecto y las pruebas se realizan en base a ello. Esto convierte a la fase de análisis en una de las más importantes y críticas del proceso de desarrollo software. En este punto ha sido fundamental el trabajo de detección de las necesidades de los clientes. Para ello se ha planteado a 14 usuarios la motivación de este proyecto y se les ha preguntado sobre lo que esperarían de la misma. Los resultados obtenidos conforman una sólida base de los requisitos que se exponen en este apartado. Se podría decir que es en este capítulo donde se asientan las bases del proyecto y, a partir de ellas, el proyecto va tomando forma.

Para empezar se ha hecho una definición de las acciones que podrá realizar cada tipo de usuario en la aplicación. Esto puede verse gracias a los casos de uso del siguiente apartado. Una vez se hayan definido las acciones y, basándonos en ellas, se detallarán los requisitos específicos de la aplicación.

5.2. Roles de usuario

El orden en el que se organizan los roles más abajo es de suma importancia, pues cada perfil de usuario incluirá las funcionalidades de los citados anteriormente incluyendolas a las que se detallen para cada rol.

- Cliente.
- Camarero.
- Administrador.

El detalle de los tipos de usuarios así como las acciones asociadas y/o restringidas para cada rol se expone más adelante.

5.3. Casos de uso

Los casos de uso definidos para el proyecto se presentan en 3 diagramas según la siguiente clasificación:

- **Diagrama de casos de uso para los usuarios sin conexión a la base de datos y/o sin escanear el código QR** (figura 5.1).

La conexión con la base de datos y la lectura del código QR son requisitos para acceder a la mayoria de las funcionalidades de la aplicación. Cumplir uno de estos requisitos sin el otro no aumenta ni disminuye el número de acciones para un usuario, ambos deben darse juntos.
También influye la conexión del dispositivo a la red local del establecimiento pero esto no le impide al usuario llevar a cabo determinadas acciones, solo implica que estas no tendrán el resultado esperado, es decir, podrá usar el chat pero no recibirá mensajes ni podrá enviarlos, también podrá elaborar solicitudes para el camarero pero éstas no serán enviadas. Estas acciones "inútiles" no se incluyen en este diagrama.
En este caso todos los usuarios se tratan de igual forma, no se hace distinción por roles.

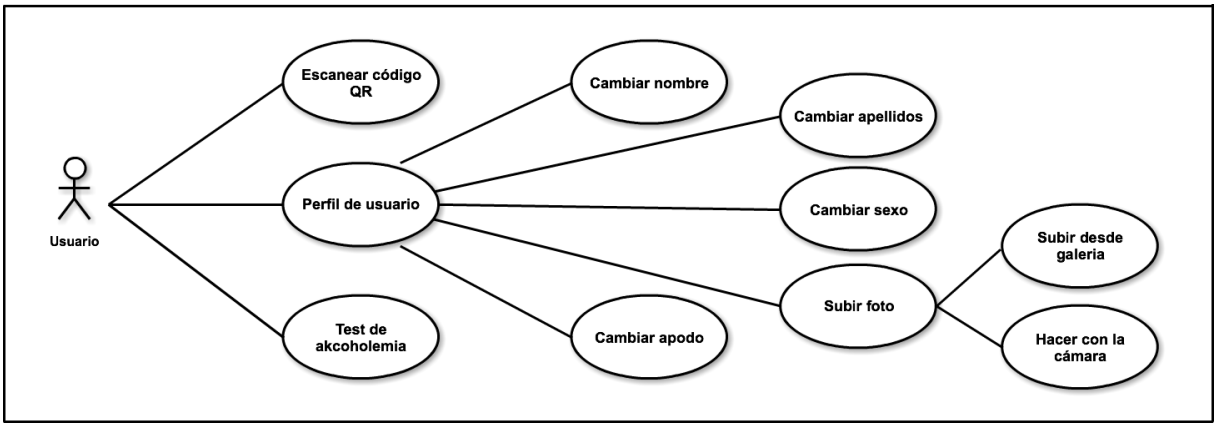


Figura 5.1: Diagrama de casos de uso para los usuarios sin conexión a la base de datos y/o sin escanear el código QR.

- **Diagrama de casos de uso para los usuarios que tienen acceso a la base de datos y han escaneado el código QR**
  - Acciones comunes a todos los usuarios (figura 5.2).
  - Trabajadores del establecimiento (figura 5.3).

5.4. Catálogo y definición de requisitos

Los requisitos definidos para esta aplicación se pueden clasificar en dos grupos:

Requisitos funcionales: especifican las funcionalidades propias del sistema.

Requisitos no funcionales: en este grupo se clasifican aquellos objetivos que no describen funcionalidades de la aplicación. Se tienen en cuenta por ejemplo, aspectos de diseño, usabilidad, rendimiento, etc.

A continuación se identifican los requisitos más relevantes para detallar la funcionalidad de la aplicación y sus características.

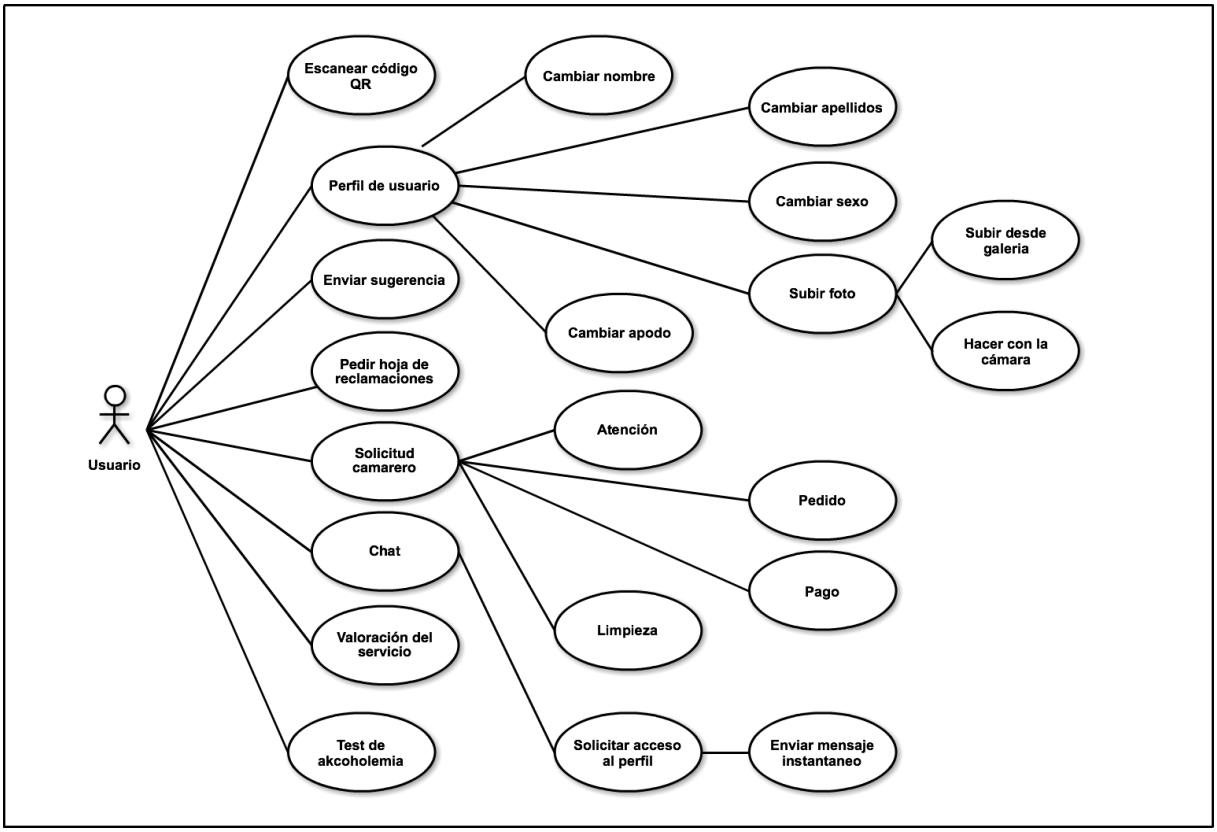


Figura 5.2: Diagrama de casos de uso para los usuarios sin conexión a la base de datos y/o sin escanear el código QR.

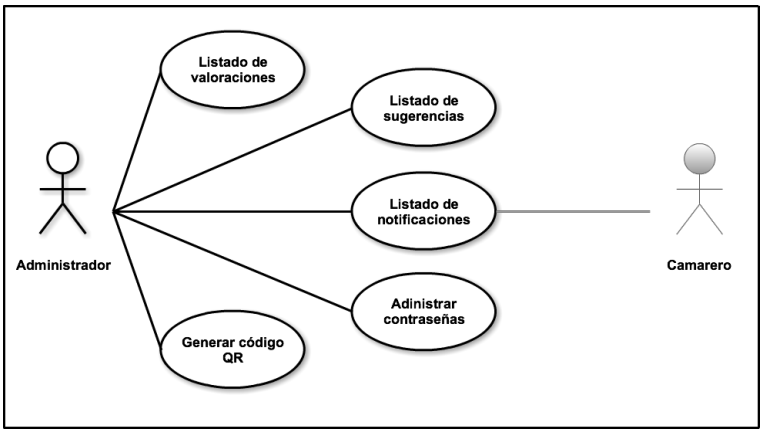


Figura 5.3: Diagrama de casos de uso para los usuarios sin conexión a la base de datos y/o sin escanear el código QR.

5.4.1. Requisitos funcionales

RF-1.– Registro. Para que el gestor del negocio pueda empezar a configurar los datos necesarios en la aplicación deberá realizar previamente un registro en la página web que se ha diseñado.

RF-2.– Lectura de códigos QR. Para que los usuarios conozcan datos del establecimiento y el establecimiento también pueda conocerlos del cliente, éste deberá escanear un código QR.
El QR le porporcionará al cliente los datos de la red local y el número de mesa para que pueda adjuntarlo en las peticiones que realice y los trabajadores sepan de donde viene cada solicitud.

RF-3.– Autenticación.

RF-3.1.– Administrador. Para acceder a las funcionalidades propias de este rol el gestor deberá indicar el identificador de su negocio y la contraseña asignada. Estos datos los devuelve la página web cuando se completa correctamente el registro. Una vez se haya efectuado el primer acceso podrá modificar todas las contraseñas.

RF-3.2.– Camarero. El acceso del camarero también requiere de una contraseña. En este caso deberá recibirla del administrador.

RF-4.– Configuración del perfil de usuario. El usuario podrá configurar los datos que serán utilizados para la comunicación interna entre clientes del establecimiento. Estarán almacenados en las preferencias [40] del teléfono. Esto permite que se mantenga la información al salir de la aplicación e incluso al apagar el dispositivo.

- **Nombre.**
- **Apellidos.**
- **Apodo.**
- **Foto.**
  - **Seleccionar de la galería.**
  - **Hacer una foto.**
- **Sexo.**

RF-5.– Peticiones al camarero. El cliente podrá realizar diferentes peticiones al camarero según la naturaleza de su necesidad.

- **Atención.**
- **Limpieza.**
- **Solicitud personalizada.**
- **Cuenta.**
- **Pago electrónico.**

RF-6.– Gestión de la prioridad de las peticiones. Una solicitud realizada por un usuario aparece en la lista de notificaciones únicamente una vez, independientemente del número de veces que se realice.

RF-7.– Prioridad. Hay 4 niveles de prioridad que aumentarán a medida que se reciban peticiones de un mismo tipo y origen. El camarero será responsable de evaluar estas prioridades como considere oportuno.

RF-8.– Carrito. El sistema permite visualizar los productos del local clasificados por tipos para hacer la búsqueda más sencilla. En la lista de productos el cliente podrá acceder a una descripción más detallada del producto, así como añadir y quitar productos al pedido.

RF-9.– Pedido. Los elementos se añaden al pedido desde el carrito y, una vez verificado por el usuario, podrá enviarselo al camarero. En el pedido también tiene opciones de gestión que le permiten añadir o quitar productos.

RF-10.– Pedido personalizado. El pedido personalizado tiene como objetivo que un cliente pueda solicitar modificaciones sobre los productos existentes o bien realizar un pedido de algún producto que no se encuentre en la carta. Las modificaciones que se pueden realizar no tienen restricciones, es decir, pueden variar desde detalles en la condimentación de la comida hasta variaciones de los elementos de un plato o un menú. El texto de esta solicitud es completamente libre.

RF-11.– Comunicación interna.

RF-11.1.– Chat general del local. En este chat pueden comunicarse todas las personas que estén conectadas a la misma red local de un establecimiento y tengan esta funcionalidad abierta en la aplicación. Desde aquí podrán solicitar una información más detallada del perfil de cualquiera de los participantes y así podrá abrir una comunicación privada.

RF-11.2.– Mensajería instantánea privada. Cuando un cliente solicita información detallada del perfil de otro y obtiene respuesta puede iniciar un hilo de mensajería instantanea a través de eventos y diálogos de la aplicación. Este hilo solo permanecerá funcional mientras un cliente reciba un mensaje y lo conteste. Si al recibir un mensaje utiliza la opción cerrar, el hilo de comunicación se cerrara a la espera de que se inicie de nuevo con una solicitud de información ampliada del perfil.

Este concepto puede resultar enrevesado pero se explica en detalle en posteriores apartados.

RF-12.– Envío de valoraciones. El cliente podrá enviar valoraciones sobre la atención y el servicio recibido durante su estancia en el establecimiento.

RF-13.– Envío de sugerencias. El cliente podrá enviar sugerencias que ayuden a mejorar los servicios que ofrece un establecimiento.

RF-14.– Solicitud de hoja de reclamaciones.

RF-15.– Test de alcoholemia. El test de alcoholemia permitirá que los clientes estimen la tasa de alcohol ingerido para garantizar su seguridad si se disponen a conducir al abandonar el establecimiento.

RF-16.– Generar códigos de acceso. Los códigos QR se utilizan para que los clientes e incluso los camareros puedan conocer datos del local que son necesarios para el correcto funcionamiento de la aplicación. El encagado de generarlos es el administrador de un negocio.

Los camareros y los clientes necesitaban prácticamente la misma información en el código pero se ha intruducido una variable que permite distinguirlos.

- **Camarero.** El número de mesa es -1.
- **Cliente.** El número de mesa es asignado por el administrador al generar el QR.

El objetivo es que cada mesa conste con un QR diferente para poder localizar las peticiones.

RF-17.– Listar las valoraciones. El administrador tendrá acceso al listado de valoraciones emitidas por los clientes de su establecimiento.

RF-18.– Listar las sugerencias. El administrador tendrá acceso al listado de sugerencias emitidas por los clientes de su establecimiento.

RF-19.– Cambiar contraseñas de acceso. El administrador podrá modificar su contraseña de acceso y la de los camareros cuando lo considere oportuno. La primera contraseña del administrador es generada aleatoriamente y devuelta por la página web tras el registro.

RF-20.– Listado de notificaciones. Entre las funcionalidades de los camareros se incluye el acceso a un listado en el que se irán añadiendo las solicitudes de los clientes en el orden de llegada. Se ha impedido la repetición de solicitudes a través de la gestión de prioridades. El camarero tendrá además una funcionalidad que le permitirá eliminar las solicitudes de la lista cuando hayan sido atendidas.

5.4.2. Requisitos no funcionales

RNF-1.– Compatibilidad. La aplicación será compatible con las APIs de Android más distribuidas en el mercado en los ultimos años. Para ser más exactos es compatible desde la 16 hasta la 23, que es la última conocida.

RNF-2.– Conexión a la red local. Los clientes necesitarán una red local en el establecimiento para comunicarse con los camareros u otros clientes. Además, esta red les proporcionará conexión a internet para acceder a la base de datos externa.

RNF-3.– Conexión internet. La conexión a internet es fundamental para acceder a la base de datos. Está debera ser proporcionada por la red local del establecimiento.

RNF-4.– Interfaz de usuario intuitiva. El número de acciones se minimiza en cada pantalla para conseguir que el usuario encuentre la información que busca fácilmente.

- **RNF-4.1.– Menús de opciones.** Puesto que no se aportan manuales con la aplicación se ha perseguido que los menús fuesen lo suficientemente intuitivos como para que los clientes entiendan en todo momento las acciones que pueden realizar a través de esta.

RNF-5.– Privacidad. Los mensajes que son lanzados a la red no son almacenados en ninguna estructura. Tampoco pasan a través de un servidor, los usuarios funcionan como clientes y servidores simultáneamente.

6

Diseño e implementación

6.1. Introducción

En esta sección se detallan los aspectos más importantes de las fases del desarrollo diseño e implementación [16, Cap. 6]. Para aportar el mayor detalle posible sobre el diseño se han definido la arquitectura general del proyecto con sus correspondientes elementos, la página web, la utilización de códigos QR, la interfaz de usuario y la estructura completa del proyecto incluyendo el diagrama de navegación de la aplicación.

Sobre estos diseños se detallará la codificación utilizada y algunas de de las decisiones más importantes tomadas sobre la implementación. El objetivo de esta codificación es hacer funcional el diseño especificado.

6.2. Arquitectura de la aplicación

Dados los requisitos del apartado anterior se ha propuesto una arquitectura para la aplicación que permite darles solución. Así ***The Waiter*** se compone de varios elementos que combinados aportan la funcionalidad planteada. La aplicación está basada en una arquitectura peer-to-peer (P2P) que permite que no existan clientes o servidores fijos, si no que todos los usuarios conectados se comporten como iguales entre sí, adquiriendo las funcionalidades de un cliente o un servidor cuando sea necesario e intercambiando continuamente estos roles.

Con esto se definen los elementos principales en los que se basa la arquitectura de la aplicación (figura 6.1):

- **Base de datos.** Para la base de datos se utiliza un modelo relacional. Está diseñada para almacenar los datos de los negocios y ponerlos a disposición de los clientes. Esto permite que los usuarios tengan acceso, por ejemplo, al listado de productos y precios del establecimiento.
- **Red.** Para el correcto funcionamiento de la aplicación los establecimientos deberán poseer dispositivos que permitan crear una red local.
- **Usuarios.** El dispositivo de cada usuario cuando se instala la aplicación se convierte en un nodo de una red multicast sobre UDP.

Para la comunicación entre los usuarios de la red se ha implementado un protocolo que se detallará en posteriores apartados. Como ya se ha mencionado, para permitir dicha comunicación es necesario que los usuarios estén conectados a una misma red local. Es decir, la aplicación está preparada para ser utilizada en multitud de establecimientos con sus correspondientes redes locales pero un usuario solo podrá comunicarse con los usuarios de la red que pertenezca al establecimiento en el que se encuentra.

La base de datos se encuentra alocada en un servidor privado de la Universidad Autónoma de Madrid por lo que puede ser accedida desde cualquier punto que tenga conexión a internet. El objetivo es que la red local

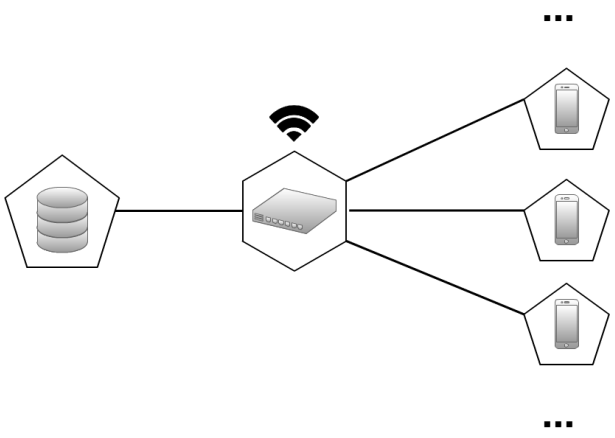


Figura 6.1: Diagrama general de la arquitectura del sistema.

proporcionada por el negocio cuente con acceso a internet puesto que de otro modo los usuarios no podrán conectarse a la base de datos al mismo tiempo que a la red del local y esto limitaría mucho la funcionalidad de la aplicación.

Para permitir que los negocios puedan hacer su primer acceso a la aplicación y configurar sus datos se ha desarrollado una pagina web alojada en el mismo servidor en el que se encuentra la base de datos. Esta herramienta no se ha mencionado como un elemento del proyecto porque su funcionalidad es de apoyo para la aplicación. En este sentido, permite unicamente hacer un registro inicial y, después de esto, toda la configuración puede hacerse desde la aplicación.

6.2.1. Usuarios

No es necesario mencionar la importancia que tienen los usuarios para este proyecto. Una vez conectados a la red local con acceso a internet y escaneando un código QR que se situaría, por ejemplo, en la mesa, los usuarios tendrán acceso a la funcionalidad completa de la aplicación.

En este apartado se detallarán aquellas partes de la estructura de la aplicación que tienen que ver con un tipo de usuario en concreto. En capítulos anteriores se han mencionado estos tipos de usuarios y ahora se detallarán sus funciones propias.

Las funcionalidades a las que tienen acceso todos los usuarios se especifican en el apartado 6.6. Aquí se resumen las funcionalidades que necesitan permisos especiales, como son las de los administradores o los camareros.

- **Administrador.**
  - Configuración de contraseñas.
  - Listado de valoraciones.
  - Listado de sugerencias.
  - Generación de códigos QR.
- **Camarero.**
  - Listado de notificaciones.

Como se ha mencionado previamente, el administrador tiene acceso a las funcionalidades del resto de roles de los usuarios de la aplicación.

Las imágenes de las pantallas correspondientes a las funcionalidades citadas anteriormente se adjuntan en los anexos junto con el resto de pantallas de la aplicación (anexo K).

6.2.2. Protocolo de comunicación

Pese a los múltiples protocolos de comunicación que existen en la actualidad se ha decidido implementar uno propio. Esto se debe principalmente a que dichos protocolos son muy completos y en este caso se necesitaba una funcionalidad tan reducida que resultó más fácil hacer un diseño que se adaptase a las necesidades del sistema objeto de este trabajo.

El protocolo define un formato de mensajes que permite establecer la comunicación entre dos puntos. Para que exista comunicación ambos extremos deben implementar tareas de envío y recepción para lanzar mensajes a la red y capturarlos.

El envío y la recepción se han implementado sobre multicast para evitar la dependencia de servidores externos, por lo que el receptor además deberá implementar un filtro para saber que mensajes están dirigidos a su dispositivo.

Implementación de las estructuras de envío y recepción:

- **Envío.** Código disponible en el anexo B.1.
- **Recepción [41].** Código disponible en el anexo C.1.

Entre el envío y la recepción tiene lugar la implementación del protocolo, como se ha dicho, para dar lugar a la comunicación.

Estructura del protocolo:

- **Clave.** La clave está separada en el mensaje por el caracter "|" e indica la naturaleza de este. Para este protocolo se han diseñado 5 tipos de claves:
    - GLOBALCHAT. Los mensajes con esta clave están destinados al chat público en el que participan todos los clientes del establecimiento.
      Puesto que la red es multicast, los mensajes enviados por un usuario también los recibe de vuelta. Es por esto que todos los mensajes del chat se muestran en el momento de la recepción.
    - CAMARERO. Los mensajes con esta clave son solicitudes de los clientes dirigidas al camarero por lo que, de ser correcto el cuerpo, se muestran automáticamente en el listado de notificaciones.
    - PERFIL. Esta clave se incorpora en los mensajes entre usuarios del chat cuando uno desea conocer el perfil completo de otro.
    - RESPUESTA_PERFIL. Este mensaje se genera si un usuario acepta compartir su ferfil con otro que así lo haya solicitado.
    - PRIVADO. Estos mensajes transportan mensajes privados entre un par de usuarios que se han puesto de acuerdo previamente con el envío de su perfil.

- **Cuerpo.** El cuerpo del mensaje es el segundo token resultante de separar la cadena por el caracter "|". Está compuesto como mínimo de otros dos parámetros separados por los caracteres ":::". Estos parámetros son interpretados por las funciones correspondientes para ejecutar las acciones oportunas. Algunos de estos parámetros son:
    - El usuario que envía el mensaje.
    - El usuario destinatario del mensaje.
    - El mensaje.
    - El número de la mesa en la que se encuentra el usuario que envía el mensaje.

  Y otros parámetros de control y configuración menos importantes. No todos los mensajes tienen la misma configuración de parámetros en el cuerpo, cada mensaje según su funcion incorpora unicamente los necesarios.

6.2.3. Base de datos

La base de datos implementada es muy sencilla puesto que uno de los objetivos de la aplicación es que los datos no queden almacenados. La estructura diseñada es la siguiente:

- **administracion.** En esta tabla se almacenan los datos del registro de un negocio realizado a través de la página web.
- **productos_x.** Esta estructura no hace referencia a una única tabla, sino a tantas como negocios estén registrados en la aplicación, siendo sustituida la x por el identificador correspondiente del negocio en la base de datos de administracion.
- **tipo_productos.** Esta tabla permite hacer una clasificación de los productos para filtrar la búsqueda aportando la mayor facilidad de localización de los productos al usuario.

Pese a su sencillez, la tabla de productos de un negocio consta de un campo foto que no ha sido sencillo de manejar. La inserción de imágenes con SQL es posible pero hartamente complicada. En su lugar se ha desarrollado un programa en Java que permite llevar a cabo esta inserción.

Diagrama entidad-relación

La base de datos utiliza 2 tablas principales y tantas auxiliares como número de negocios se encuentren registrados. El funcionamiento de cada una de estas estructuras se ha detallado previamente, en este apartado se adjunta un diagrama para facilitar la comprensión del diseño y las relaciones que existen entre las estructuras mencionadas.

Este diagrama se puede observar en la figura 6.2.

Conexión

La conexión con la base de datos se ha realizado a través de clases Java para el trabajo con conexiones SQL [42]. La base de datos está alojada en un servidor de la Universidad Autónoma de Madrid y se accede a ella de forma remota desde los dispositivos. Por seguridad de los datos, al implantar la aplicación en un negocio real la base de datos se alojaría en modo local o al menos se realizarían copias con frecuencia de la misma.

El código utilizadopara establecer la conexión se encuentra disponible en el anexo E.

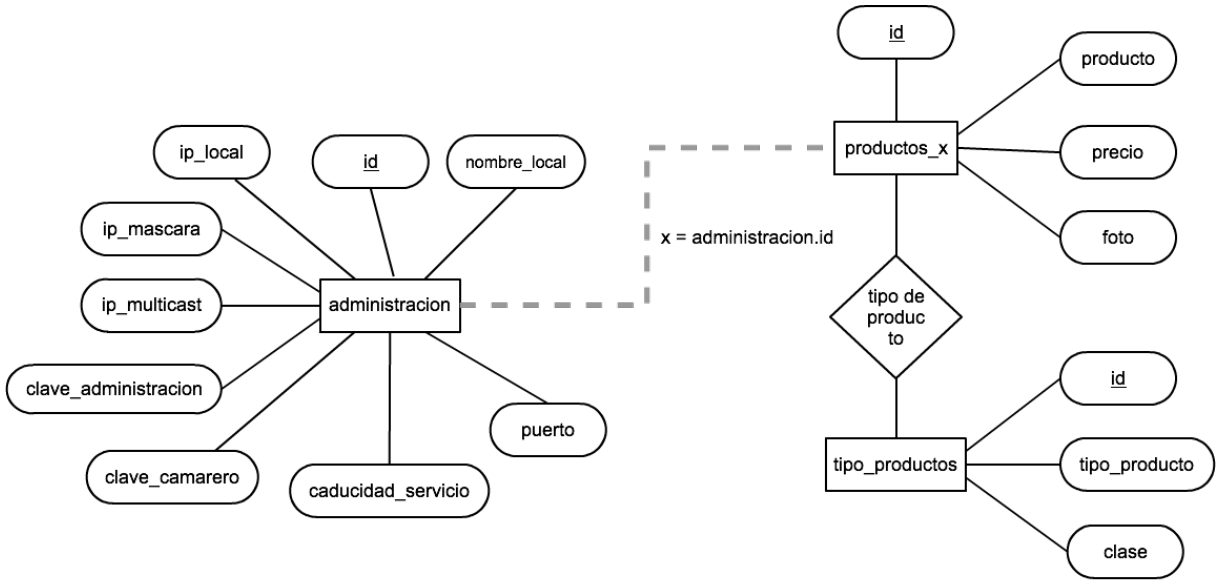


Figura 6.2: Diagrama Entidad-Relación de la base de datos.

6.3. Página web

Pese a que no es uno de los elementos principales de la arquitectura se cita en primer lugar porque compone el primer paso para comenzar a usar la aplicación.

La página web consta de 2 pantallas. En la primera se proporciona un formulario para registrar un negocio. Una vez los datos del formulario se han validado, son almacenados en la base de datos y se usa la segunda pantalla para proporcionar la información de acceso a la aplicación. Las contraseñas podrán ser modificadas pero hay un dato que no puede modificarse y debe ser conocido para poder acceder a la aplicación y administrarla, el identificador del local en la base de datos.

En el formulario se introduce el nombre del local, datos de la red y un correo electrónico para una tarea futura que consistirá en mandar la confirmación por correo electrónico.

En la confirmación, a través de la página, se proporcionan datos que serán requeridos por la aplicación para acceder como administrador: el identificador en la base de datos y una contraseña generada aleatoriamente con el fin de que en el primer acceso sea cambiada del mismo modo que la de los camareros.

Ambas pantallas pueden observarse en el anexo D. Únicamente se adjunta el trozo de la páginas donde pueden observarse el formulario y la confirmación, la página web puede ser accedida en el siguiente enlace: http://metis.ii.uam.es/ lorena/.

6.4. Códigos QR

Los códigos QR se han utilizado para que la aplicación tenga acceso a determinados datos. Estos códigos se generan desde la aplicación con permisos de administrador y sirven para que los camareros y los clientes conozcan datos sobre la red, carta del establecimiento, etc. A continuación se explican más en detalle los datos

que contiene cada código y con qué objetivo.

Para completar la funcionalidad entre la aplicación y los códigos han sido necesarios dos elementos:

- **Generador [43].** Este elemento utiliza una matriz de bits que se colorean en blanco o negro según lo indique una condición. Esta condición resulta de codificar los datos con una clase especial de Java. El código utilizado está disponible en el anexo G.
  Aunque no es una funcionalidad prevista para este proyecto, en lugar de usar una herramienta existente se decidió implementar en parte por su simplicidad, como puede observarse en el código.
- **Lector [44].** El lector utilizado es una herramienta Android existente con la que se comunica ***The Waiter*** cuando es necesario. Esta aplicación se llama Barcode Scanner [45]. ***The Waiter*** comprueba si esta alicación está instsalada cuando se quiere escanear un código y si no lo está dirige a los usuarios a la tienda para que puedan descargarla.

La información que es almacenada en los códigos QR es la siguiente:

- **Clave de verificación.** Con esta clave se comprueba, al leer el código, que ha sido un código válido generado por la aplicación.
- **Número de mesa.** -1 en caso de tratarse de un QR para los empleados del negocio.
- **Nombre de la red local.**
- **Contraseña de la red local.**
- **Nombre de la red local.**
- **Dirección IP.**
- **Nombre del local.**
- **Puerto de la base de datos.**
- **Dirección de la máscara de red.**
- **Dirección multicast de la red.**
- **Identificador del negocio en la base de datos.**
- **Contraseña del camarero.**

(a) Camarero

(b) Cliente

Figura 6.3: Códigos QR generados por la aplicación con el código del anexo G

6.5. Interfaz de usuario

La interfaz de usuario diseñada no sigue ninguna plantilla ni se han utilizado herramientas para su elaboración. Es una interfaz sencilla puesto que uno de los requisitos para este proyecto era proporcionar la mayor facilidad de uso a todos sus usuarios.

Siguiendo el estandar de estilo de Android, para mantener cierta coherencia entre todas las pantallas, se han utilizado estilos, declarados en un fichero llamado styles.xml en el proyecto Android. Estos estilos permiten agrupar elementos en torno a un mismo diseño, es decir, por ejemplo, permiten que todos los botones sean iguales. Estos estilos se han aplicado en la mayoría, si no en su totalidad, de los elementos de la aplicación aportando como decíamos coherencia y sencillez en todas las pantallas de la aplicación.

En anteriores apartados y en concreto en el siguiente se puede observar esta interfaz.

6.6. Estructura de la aplicación

Se podría decir que cada una de las pantallas que componen la aplicación son una actividad. Existen algunas variaciones pero en este caso no se han utilizado por lo que se tratará cada pantalla como una actividad.

Las actividades están formadas por dos partes: la parte lógica y la parte gráfica.

La parte lógica es una archivo .java que es la clase que se crea para poder manipular, interactuar y colocar el código de esa actividad.

Dicho esto, aunque la aplicación tenga como base un lenguaje orientado a objetos, Java en el caso que nos concierne, el diagrama de clases carece de importancia frente al diagrama de actividades, que se detalla en este apartado (figura 6.4). Tambien influye el hecho de que el diagrama de clases para este proyecto constaría de más de 40 clases y, más que ayudar a visualizar la estructura, dificulta el entendimiento de la misma.

En este diagrama se muestran las pantallas a las que tienen acceso todos los usuarios. Las pantallas que se han diseñado con funcionalidades adicionales para camareros y administradores se han detallado en el apartado 6.2.1.

Detalle del flujo de la figura 6.4:

1.– **Pantalla principal.** Esta es la pantalla inicial de la aplicación. Pulsando el botón “Entrar” los usuarios pueden acceder al menú principal. En el menú desplegable se ofrecen opciones de acceso a los usuarios con rol de camarero y administración y también se ofrece la opción de escanear el código QR.

2.– **Menú principal.** Desde el menú principal se puede acceder a la mayor parte de las funcionalidades comunes de los usuarios. En el menú desplegable se ofrecen las opciones de configuración del perfil de usuario, escaneo del código QR, envío de sugerencias y petición de hoja de reclamaciones. Las opciones principales de esta pantalla se muestran en el diagrama.

3.– **Configuración del perfil.** Con esta opción los usuarios pueden configurar los datos que verán otros usuarios cuando utilicen el chat o la mensajería privada.

4.– **Escaner de QR.** Una vez escaneado el QR, si es correcto, la aplicación navegará a una pantalla que informará al usuario del nombre del establecimiento en el que se encuentra y del número que tiene asignado su mesa.

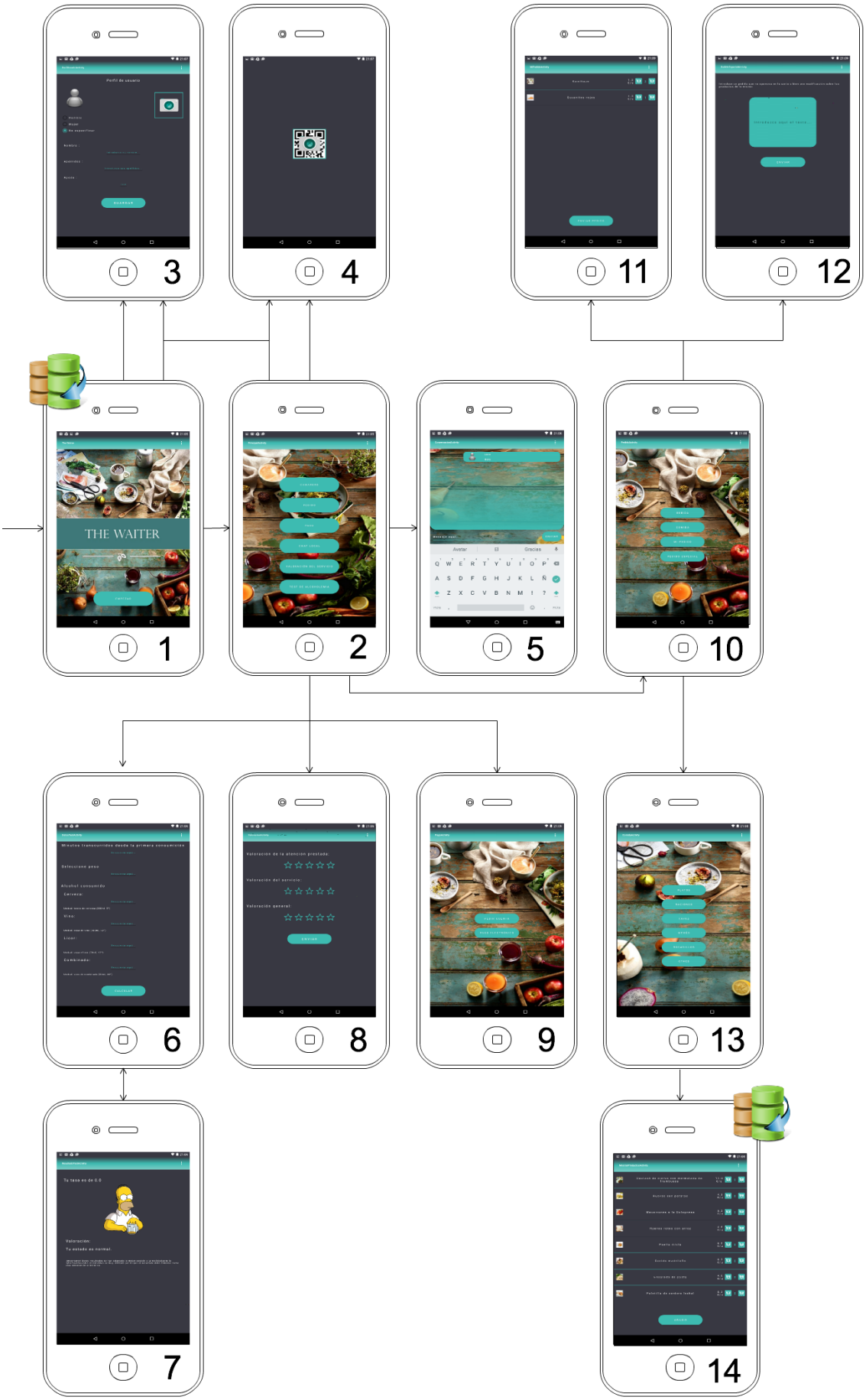


Figura 6.4: Diagrama de actividades.

5.– **Chat.** Desde el chat se puede iniciar un hilo de mensajería instantánea como está detallado en el apartado 6.6.2.

6.– **Test de alcoholemia.** En esta pantalla se ofrece un formulario para obtener los datos necesarios que permiten estimar la tasa de alcoholemia.

7.– **Resultado del test de alcoholemia.** En esta pantalla se muestra el resultado de la tasa con una breve descripción del estado del usuario correspondiente a esta tasa.

8.– **Valoración del servicio.** Permite puntuar determinados aspectos del establecimiento y enviar estos datos.

9.– **Solicitud de pago.** Se envía una solicitud al camarero de entre 2 opciones, pago en efectivo o pago electrónico, como se muestra en el menú de esta pantalla.

10.– **Pedido.** Desde la pantalla de pedido se puede acceder a la carta a través de varios menús y pantallas que han permitido clasificarla para agilizar la búsqueda de productos. También se puede acceder al detalle del pedido de un usuario y a un formulario para solicitar un pedido especial.

11.– **Mi pedido.** Aquí se muestra el detalle del pedido de cada usuario y se permite que lo envíe al camarero cuando lo estime oportuno.

12.– **Solicitud de pedido especial.** Se muestra un formulario para que el usuario introduzca un texto libre indicando un pedido con productos que no se encuentran en la carta.

13.– **Clasificación de la carta.** Hay 3 niveles de menús que permiten clasificar la carta por categorías de productos para agilizar la búsqueda de los usuarios.

14.– **Carrito.** Al carrito se accede desde los menús de clasificación de carta y en el se pueden añadir los productos listados al pedido.

La comunicación entre los clientes y el camarero así como la mensajería instantánea entre clientes no están ligadas a una única pantalla por lo se han separado de este diagrama y se describen a continuación:

6.6.1. Peticiones

El usuario puede realizar peticiones desde varias pantallas según la naturaleza de la petición. El camarero las recibirá todas en el listado de notificaciones. En el diagrama del anexo H (disponible en los anexos) se pueden observar algunas de estas peticiones. No se han incluido todas las posibilidades para no dificultar la lectura del diagrama. También se incluye un pequeño detalle sobre el flujo que se muestra en la figura.

6.6.2. Mensajería instantánea

El servicio que se aporta de mensajería instantánea está implementado con eventos. Todas las pantallas de la aplicación están diseñadas para actuar ante la recepción de estos eventos y es por esto que se ha separado este flujo, ya que no forma parte de la navegación normal de la aplicación. En el anexo I se aporta un diagrama de esta estructura de mensajería así como un pequeño detalle del flujo.

Con esto queda completa la estructura de la aplicación.

Las imágenes de la aplicación que se han utilizado para los diagramas de este apartado se encuentran en el anexo K con la finalidad de que el lector pueda observarlas mejor en un tamaño más grande.

7

Pruebas

7.1. Alcance de las pruebas

Las pruebas se han ido desarrollando a medida que se cerraban las tareas o se estimaba necesario tras las revisiones. El motivo por el que no se ha dedicado una fase aislada del desarrollo para las pruebas se basa en la escasez de recursos humanos. Así mismo, al no conocer previamente herramientas fundamentales que se han utilizado en la implementación ha resultado complicado planificar unas pruebas desde el inicio.

Es este apartado se listarán las pruebas que se han utilizado para validar y verificar el correcto funcionamiento de la aplicación.

Las pruebas realizadas se han centrado en la comprobación de la funcionalidad, compatibilidad, accesibilidad y usabilidad.

7.1.1. Funcionalidad

Las pruebas sobre la funcionalidad tienen como objetivo la comprobación del correcto cumplimiento de los requisitos funcionales planteados para el proyecto.

Para ello se han utilizado como guía los requisitos de la aplicación y se ha chequeado el correcto funcionamiento de cada uno individualmente y en contexto con otras funcionalidades relacionadas. Estas pruebas se han realizado en varias etapas del desarrollo a medida que se implementaban nuevas funcionalidades o se integraban nuevos elementos. Los resultados obtenidos en las últimas pruebas garantizan el correcto funcionamiento de aproximádamente un 96 % de los requisitos planteados. Esto se debe a que el requisito funcional 1(RF1) no puede ser validado completamente como se verá a lo largo del capítulo.

Para ello han colaborado 4 usuarios con conocimientos técnicos y de la aplicación, reportando un total de 34 errores que han podido ser solventados. Algunos de estos errores son:

- Unicidad de los usuarios. Al no estar registrados en una base de datos es difícil distinguirlos y por ello se ha asociado un UUID a cada usuario.
- Introducción de datos erróneos en los formularios.
- Fallos de conexión en la red.
- Acciones no disponibles sin el informe correspondiente para el usuario.
- Duplicidad en las notificaciones que reciben los camareros, lo que se ha controlado con la gestión de prioridades.

7.1.2. Compatibilidad

La compatibilidad en Android se centra en las versiones. Es dificil asegurar la compatibilidad dado que poco después de empezar el proyecto se introdujo en el mercado una nueva versión de Android, Marshmallow (versión 6.0). Con ello se han quedado obsoletas muchas funciones de Android y requiere un constante esfuerzo mantener la retrocompatibilidad y adaptarse a las actualizaciones de la nueva versión.

También hay que tener el cuenta que, desarrollando para la última version que se conoce, se ha adquirido el riesgo de que muchos dispositivos que no puedan ser actualizados no tengan acceso a la aplicación pero la decisión que se ha tomado aseguraba la mayor compatibilidad para el mayor número de usuarios posible.

La aplicación ha sido desarrollada en base a la API 16 y desde ahi hasta la 23, la cual es la más reciente a fecha de impresión de este documento. Como se ha mencionado, esta es la opción que más compatibilidad proporcionaba.

Pese a esto, como se ha dicho, la compatibilidad es muy dificil de asegurar y deja mucho trabajo para realizar en la fase de mantenimiento.

Las pruebas sobre la compatibilidad se han realizado utilizando 13 dispositivos de 7 versiones diferentes de Android. El resultado ha sido peor de lo esperado, los fallos de compatibilidad son frecuentes al compilar la aplicación para más de 5 versiones diferentes y en especial se han dado problemas con la última versión de Android, Marshmallow (versión 6.0, API 23).

7.1.3. Accesibilidad

Se realizan pruebas de este tipo para garantizar el correcto acceso de todos usuarios a aquellas funcionalidades que estén a su disposición en función de los roles y los permisos de cada uno.

Estas pruebas se han basado en comprobar que el usuario accede a todas las funcionalidades de la aplicación de forma intuitiva y que únicamente accede a aquellas cuyo rol le permite. Para ello se han utilizado 9 usuarios con las instrucciones de usar la aplicación e informar de todas las funcionalidades que encontrasen en ella. El resultado ha sido favorable, no se han detectado errores en los accesos dependientes de los permisos del tipo de usuario y de media han encontrado el 94,3 % de las funcionalidades de la aplicación.

7.1.4. Usabilidad

Se realizan pruebas sobre la usabilidad puesto que a través de ellas se obtiene una información directa irremplazable de como los usuarios reales utilizan el sistema.

Estas pruebas se enfocan en medir la capacidad de un producto para cumplir el propósito por el cual fue diseñado.

Métricas

Las métricas sobre la usabilidad que se han utilizado son las siguientes:

- **Exactitud.** El número de errores cometidos por los usuarios al utilizar el sistema fue muy reducido y en todos los casos fueror recuperables al usar los datos o procedimientos adecuados.
- **Tiempo.** El tiempo medio requerido para realizar cualquiera de las acciones que permite la aplicación es de 5 segundos.

- **Recuerdo.** Los usuarios tras un periodo medio de dos meses sin utilizar la aplicación recuerdan las funcionalidades más accedidas en media, es decir, el QR se escanea únicamente una vez pero las peticiones al camarero pueden ser muchas en cada visita, estas acciones por lo tanto son las más recordadas, por ser las más utilizadas en media.
- **Respuesta emocional.** Los usuarios han reportado un grado elevado de satisfaccion al realizar cualquiera de las tareas posibles de la aplicación. También han mostrado entusiasmo ante la idea de implantar el sistema en un negocio real.

Características del sistema

The Waiter se basa en los criterios citados anteriormente, consiguiendo así una herramienta fácil de aprender y usar, flexible, consistente, que proporciona un buen tiempo de respuesta, se adecúa a las tareas y disminuye la carga cognitiva.

7.2. Estrategia y desarrollo

7.2.1. Estrategia

Se has realizado pruebas unitarias sobre todos los elementos de la aplicación. En objetivo de estas pruebas ha sido conformar una estrategia de integración de dichos elementos en el proyecto. Una vez incorporados los elementos a la aplicación se han realizado las pruebas correspondientes de integración que en último lugar han dado lugar a las pruebas generales del sistema. La estrategia de integración que se ha utilizado consiste en dar preferencia a aquellos elementos con mayor número de funcionalidades básicas y que, por lo tanto, serían necesarios para elementos incorporados posteriormente. Cuando este criterio no ha resultado suficiente para seguir un orden en la incorporación de nuevos elementos, se ha tenido en cuenta si formaban parte de una funcionalidad principal o secundaria, es decir, si influian en el objetivo del proyecto o eran aportaciones de apoyo al objetivo principal.

7.2.2. Tipo de pruebas realizadas

Pese a la distinta naturaleza de los elementos que componen la aplicación, el tipo de pruebas realizadas es el mismo. Esto ha supuesto simplicidad en la etapa de pruebas destinando en la medida de los posible el mayor número de horas a la etapa de desarrollo.

Las pruebas que hemos utilizado son las conocidas como pruebas de caja gris (caja blanca + caja negra):

- Caja blanca. Estas pruebas se han realizado teniendo en cuenta la estructura interna de la aplicación. Su principal objetivo es explorar todas las posibilidades del flujo de la aplicación que se pueden dar en el código.
- Caja negra. En estas pruebas no se tiene en cuenta la estructura interna de la aplicación por lo que nos centramos unicamente en las salidas que produce la aplicación ante determinadas entradas.

En este proyecto no se han realizado pruebas de caja blanca y caja negra como tal, sino que por el escaso número de desarrolladores se han realizado pruebas de caja gris. Estas pruebas consisten en una base de pruebas de caja negra centradas en puntos conflictivos que son conocidos por las personas que conocen bien la

estructura interna del programa. Estas pruebas pueden resultar muy efectivas en el sentido de que se conocen bien aquellas partes de la aplicación que pueden resultar mas problemáticas.

La aplicación se ha probado exhaustivamente por los desarrolladores para comprobar el correcto funcionamiento de cada uno de los requisitos. Puesto que el equipo de desarrollo cuenta con una única persona, las pruebas también han sido realizadas por usuarios con conocimientos de informática y sin ellos. Para aquellos usuarios que no tenían conocimientos de informática no se han aportado instrucciones puesto que una de las pruebas más importantes consistía en medir la facilidad e intuitividad que se quería para la aplicación.

Las pruebas más destacables que se han realizado sobre la aplicación son las siguientes:

- Control de accesos. Comprobación de los accesos a determinadas funcionalidades en función del rol que ostente el usuario. Los accesos también se han limitado según los datos conocidos por la aplicación, es decir, sin la información del QR o de la base de datos no se ha permitido el acceso a funcionalidades que requieren de esta información.
- Interfaz de usuario. En estas pruebas se ha perseguido la comodidad de los usuarios haciendoles en todo momento accesibles las funcionalidades necesarias de la aplicación. Basicamente se ha perseguido una navegación sencilla para el usuario por las pantallas de la aplicación.
- Conexión. Uno de los puntos más problematicos es el protocolo de comunicación ya que al ser UDP puede existir pérdida de mensajes. Para evitar esto se requiere de una recepción de señal intensa de la red para los usuarios y es algo que no se ha controlado internamente desde la aplicación.

La aplicación no se ha podido someter a pruebas en un negocio real pero si se ha intentado en lo posible simular circunstancias parecidas. Las pruebas de integración se han realizado en el entorno de desarrollo.

7.3. Resultados y conclusiones

Los resultados obtenidos en estas pruebas validan un 96 % de los requisitos funcionales plantedos para este proyecto. Los fallos que no permiten completar el correcto funcionamiento de todos los requisitos han sido detectados pero no solventados.

Los fallos de compatibilidad requieren una inspección constante de las funcionalidades de Android, tanto las que hayan sido declaradas como obsoletas como las que hayan surgido en solución a esto. Las versiones tienen una media de 2 APIs y cambian, en media, 2 veces al por lo que no ha sido posible dar soporte a más de 5 versiones consecutivas aunque inicialmente se plantearon 8.

La red local sufre constantes perdidas de paquetes debidas al hardware utilizado. Esto se ha solventado mejorando las prestaciones del hardware utilizado para la red local. A nivel de aplicación no ha podido solventarse puesto que son elementos externos a la misma.

Resumiendo lo expuesto en los apartados anteriores los resultados han sido, en general, son buenos. Se ha conseguido que los usuarios utilicen la herramienta de forma intuitiva, los requisitos planteados se cumplen al 96 % utilizando un router que proporcione una alta intensidad de señal en la red local, el acceso de los diferentes tipos de usuarios a diferentes funcionalidades se ha validado correctamente y se han corregido todos los errores reportados por los usuarios obteniendo finalmente opiniones muy positivas sobre ***The Waiter***.

8 Evaluación

8.1. Evaluación de los usuarios

The Waiter se ha sometido a diversos procesos de evaluación. La evaluación realizada por el equipo de desarrollo ha sido exahustiva pero no suficiente debido a que tener un amplio conocimiento de la aplicación impide descubrir en muchos casos carencias o incluso fallos de la misma. Es por esto que ***The Waiter*** ha sido evaluado por un grupo de 17 usuarios externos al equipo de desarrollo. Estos usuarios tenían conocimientos muy dispares sobre el lenguaje de desarrollo, las tecnologías y herramientas utilizadas e incluso sobre la utilidad del proyecto.

Uno de los principales objetivos, como se ha comentado en capítulos anteriores, es el desarrollo de una interfaz intuitiva y amigable de modo que los usuarios no requieran un manual o información previa sobre la aplicación. En cambio los negocios que quieran utilizar la alicación si necesitarán entrar en contacto con los desarrolladores pero unicamente para dar los primeros pasos tales como el registro y la configuración inicial.

En general, los usuarios han reportado multitud de errores y sugeriancias que han enriquecido la funcionalidad de la aplicación. La mayoría de los errores eran prácticamente inadvertibles por el equipo de desarrollo por su aparición en casos extremos e incluso irreales, como, por ejemplo, la posibilidad de introducir cantidades negativas. A raiz de un fallo en la mayoría de las ocasiones se han detectado otros de la misma natiraleza. Pese a esto, fallos en situaciones extremas como las citadas previamente se salen del flujo de la aplicación y solo prodrían ser enmendados en su totalidad con la colaboración de un usuario experto.

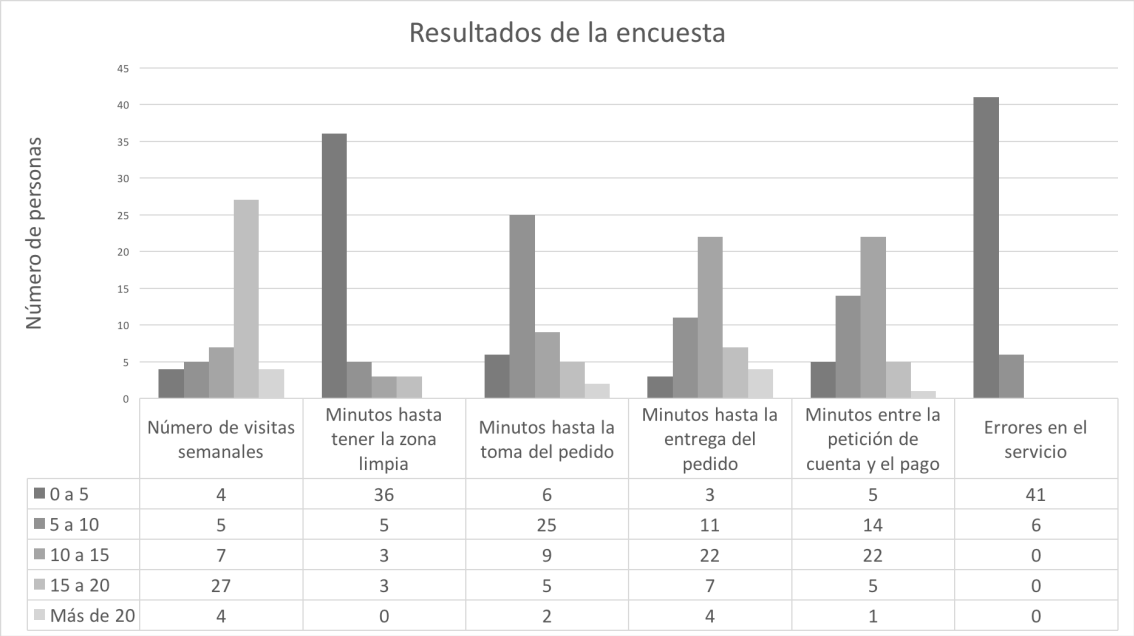
En cuanto a la interfaz y la usabilidad, los usuarios no han reportado apenas errores, sino todo lo contrario, han destacado de manera muy positiva el diseño y la usabilidad de la interfaz.

A parte de realizar una evaluación sobre la aplicación era fundamental conocer el impacto que ***The Waiter*** podría causar en el mercado actual, incluso si las carencias de los negocios por las que comenzó este proyecto eran compartidas por otros usuarios. Para ello, se ha elaborado una encuesta que ha sido respondida de forma válida por 47 usuarios. La encuesta está disponible en el anexo J.1. El detalle de los resultados se ha elaborado como un diagrama (figura 8.1).

Con los datos obtenidos en esta encuenta se ve confirmada la realidad de que los tiempos de servicio que se dan en la actualidad son elevados. Por lo que se ve justificada la necesidad este proyecto.

8.2. Beneficios de la aplicación

Los beneficios que ***The Waiter*** puede proporcionar a los negocios destinados a la hostelería y a los usuarios de estos son muchos, la mayoría se han ido comentando a lo largo del documento. Incluir esta herramienta en en los locales y los dispositivos de los clientes supondrá un método de agilización de las peticiones que los clientes



	Número de visitas semanales	Minutos hasta tener la zona limpia	Minutos hasta la toma del pedido	Minutos hasta la entrega del pedido	Minutos entre la petición de cuenta y el pago	Errores en el servicio
0 a 5	4	36	6	3	5	41
5 a 10	5	5	25	11	14	6
10 a 15	7	3	9	22	22	0
15 a 20	27	3	5	7	5	0
Más de 20	4	0	2	4	1	0

Figura 8.1: En este diagrama se detallan los resultados obtenidos al realizar la encuesta a 47 usuarios, 5 de los cuales afirman que sus datos se ven alterados por coincidir con fechas especiales.

realizan a los camareros. También servirá como plataforma para establecer una comunicación entre todos los usuarios de la aplicación que se encuentren conectados a la red local del establecimiento.

Uno de los beneficios más importantes y una de las razones por las que se decidió crear este sistema, es para que los clientes de un negocio puedan procesar sus solicitudes cuando así lo deseen, sin necesidad de que un camarero les mire al pasar, atienda su llamada o acuda a su mesa. Esto potencia en los clientes cierta sensación de autonomia y aumenta su satisfacción sobre la estancia en el establecimiento. Sin olvidar que esto supone una gran mejora para los camareros que podrán recibir todas las peticiones en un dispositivo móvil sin necesidad de ir mesa por mesa primero para conocer la naturaleza de la petición y después para atenderla. Sin cuadernos y bolis, una única herramienta al alcance de todos.

9

Conclusiones y líneas futuras

9.1. Conclusiones

En este trabajo se ha desarrollado y presentado una herramienta para dispositivos móviles y tabletas que permite reducir el tiempo de estancia por esperas “innecesarias” en los establecimientos decicados a ofrecer servicios de hostelería, más en concreto servicios de alimentación. Duante este proceso ha sido fundamental el trabajo de detección de las necesidades de los clientes y los establecimientos para cumplir con el objetivo principal, elaborar una herramienta “útil”, que solucione necesidades de los usuarios.

Una de las conclusiones derivadas de la experiencia con los usuarios es que sería una herramienta de gran utilidad por los bajos costes que supone implantar el sistema en un establecimiento y la comodidad que proporciona a los usuarios por ambos partes, a los clientes y al negocio. Pese a la gran acogida que ha recibido por parte de los usuarios que la han conocido y probado, se han recibido multitud de sugerencias que permitirían hacer más completas las opciones que ofrece la aplicación. Por la limitación de tiempo, y en algunos casos, de recursos estas mejoras se han propuesto como líneas de trabajo futuro.

Las conclusiones finales del presente Trabajo de Fin de Grado son las siguientes:

- Se ha desarrollado una herramienta eficaz de comunicación entre los clientes de un establecimiento y sus trabajadores.
- La herramienta proporciona una comunicación privada sin el uso de servidores entre los usuarios del establecimiento a traves de un servicio de chat y otro de mensajería instantánea entre dos usuarios que para ser iniciado requiere la aceptacion de las dos partes.
- La herramienta que se presenta duplica la funcionalidad que estaba prevista en un primer momento.
- La interfaz que se ha diseñado se ha orientado, en todo momento, a la facilidad de manejo y entendimiento por parte de los usuarios. Se ha conseguido una interfaz sencilla e intuitiva que cumple los estándares de Android y se ajusta a un gran número de criterios de usabilidad.
- Se ha desarrollado para ser soportada por la gran mayoría de los dispositivos Android que puedan estar actualmente en uso, aquellos cuya versión esté entre la 4.1 y la 5.0 puesto que estas versiones son las más distribuidas como se puede observar en el gráfico del anexo A.
- Se ha priorizado que los elementos que componen la herramienta así como el conjunto final sean reutilizables y ampliables de una forma sencilla.

9.2. Trabajo futuro

9.2.1. Modularidad

Como se ha comentando en secciones anteriores del documento, uno de los objetivos del desarrollo era poder extraer funcionalidades para construir así aplicaciones completamente distintas a la que se presenta.

En este sentido, se han planteado algunas de las posibles opciones que podrían existir al desligar las funcionalidades que conforman ***The Waiter***:

- Chat. Una de las funcionalidades que presenta el proyecto es un chat abierto para todos los usuarios de la red local. Las ventajas que puede presentar con respecto a otros servicios existentes de mensajería instantánea es que carece de servidores y la información que se intercambia no es almacenada. Es más, ni siquiera las conversaciones de un usuario serán almacenadas una vez las abandone.
- Generador de códigos QR. Este elemento al igual que el chat no formaba parte explícita de los objetivos de la aplicación pero si ha sido necesario para cumplirlos. Esto permite que la aplicación obtenga infomación acerca del establecimiento en el que se encuentre el cliente. Existen multitud de generadores de códigos QR y éste sin modificaciones simplemente sería uno más por lo que no sería competitivo en el mercado pero con algunas mejoras no contempladas en el alcance de este proyecto podría serlo.
- Test de alcoholemia. En la aplicación se presenta un test de alcoholemia que tiene como objetivo garantizar en la medida de lo posible la seguridad de aquellos clientes que consuman bebidas alcoholicas en el establecimiento. Los cálculos que se aplican para calcular la tasa son fiables pero el resultado no lo es tanto. Esto tiene que ver con el nivel de veracidad de los datos que introduzca el usuario. Aun siendo todos los datos correctos, el resultado que ofrece esta funcionalidad debe tomarse como algo meramente orientativo. Basándonos en las pruebas realizadas con usuarios se ha puesto en duda que se utilice para fines de seguridad, en su lugar se cree que se utilizará más como divertimento.

Como se ha dicho previamente, estas funcionalidades son algunas que se han planteado pero pueden existir multitud de usos más, incluso utilizando combinaciones de las funcionalidades que se desliguen de la aplicación.

9.2.2. Extensibilidad

La extensibilidad ha sido una de las metas a alcanzar en todo el desarrollo del proyecto, debido a que se ha planteado como un producto software abierto a cualquier tipo de mejora. Esto significa que, debido en gran parte a la modularidad de la misma, la aplicación podría incrementar su tamaño, carga y arquitectura sin necesidad de manipular total o parcialmente la mayoría de componentes implementados y así poder extender la funcionalidad añadiendo una capa más al software de base.

Por esto mismo, se puede decir que la aplicación presenta extensibilidad como un todo. Para cubrir todas las necesidades de los negocios de hostelería se implementaría otra aplicación que trabaje conjuntamente con ***The Waiter*** y realice las funcionalidades básicas de un TPV. Esto se observa mejor en el siguiente apartado en el que se detallan las líneas futuras del proyecto.

9.2.3. Líneas futuras

Las líneas de trabajo futuro que se han propuesto para el proyecto son las siguientes:

- Ampliar la funcionalidad destinada al negocio implementando una herramienta compatible con ***The Waiter*** para dispositivos móviles que de solución a las necesidades que actualmente cubren los negocios con los TPV.
- Ampliar las funcionalidades que tiene el administrador para configurar todos los datos referentes al establecimiento desde la aplcación.
    - Configuración de los productos de la carta.
    - Modificación de los datos del registro inicial, tales como el nombre del local o la dirección IP.
    - Impresión de los códigos QR generados desde la aplicación.
- Incluir pasarelas de pago para completar la opción de pago electrónico de los clientes.
- Gestión de socios. Incorporar un código QR impreso en la cuenta que permita premiar a los usuarios por su fidelidad con el negocio acumulando, por ejemplo, descuentos.
- Actualmente la mensajería instantánea entre dos usuarios no permite acumular mensajes anteriores, lo que evita que exista un contexdo de la conversación. Puesto que se ha trabajado sin servidores y es un aspecto que no pretende cambiarse, para almacenar el contexto de las conversaciones se utilizarán ficheros temporales en el dispositivo o bien se hará uso de la base de datos SQLite que ya está incorporada en los dipositivos Android.
- Ampliar la funcionalidad de la página web para que los administradores puedan configurar los datos del negocio bien desde la aplicación o bien desde la página.
- Puesto que se trabaja sobre UDP, la comunicación puede sufrir pérdida de mensajes. Una de las opciones es implementar un servicio de control de entregas y otra trabajar sobre un protocolo que ya lo garantice, como por ejemplo, TCP.
- Implementación de otras opciones para los usuarios. Estas ideas son fruto de las sugerencias que se han recibido durante la evaluación de la aplicación.
    - Peticiones rápidas: salero, salsas, cubiertos, servilletas.
    - Incluir un buscador de productos.
    - Recepción de notificaciones: mensajes, pedido recibido, etc.

Como se ha explicado, el proyecto se puede ampliar de muchas formas que permitan hacer cada vez más completa y útil la herramienta presentada aunque el trabajo presentado cumple con creces los objetivos planteados al inicio del proyecto.

Bibliografía

[1] J. R. Lequerica, *Desarrollo de aplicaciones para Android*. Anaya, 2016.

[2] W. F. Ableson, R. Sen, and C. King, *Android. Guía para desarrolladores*, 2nd ed. Anaya.

[3] J. E. A. Soriano, *El gran libro de programación avanzada con Android*. Marcombo, 2012.

[4] J. Tomás, *El gran libro de Android*, 5th ed. Marcombo, 2016.

[5] "Android SDK," https://es.wikipedia.org/wiki/Desarrollo_de_programas_para_Android%#Android_SDK, accedido: 14/10/2015.

[6] R. Cadenhead, *Java 8*. Anaya.

[7] P. A. Sznajdleder, *Java A Fondo. Estudio Del Lenguaje Y Desarrollo De Aplicaciones*. Marcombo.

[8] B. Eckel, *Piensa en Java*. Pearson.

[9] "Custom adapter implementations," http://www.vogella.com/tutorials/AndroidListView/article.html#adapterown_example, accedido: 27/10/2015.

[10] "Android: Elementos de ListView con varios botones cliqueables," https://datafull.co/p/android-elementos-de-listview-con-varios-botones-cliqueables, accedido: 17/02/2016.

[11] "Android: duplicar los elementos en ListVew. Quizás getView() llama demasiadas veces?" http://es.androids.help/q24687, accedido: 14/02/2016.

[12] "Programación Android: Interfaz gráfica – Adapters I," https://elbauldelprogramador.com/programacion-android-interfaz-grafica%_28/, accedido: 21/03/2016.

[13] "Build an Android Instant Messaging App Using Sinch and Parse," https://www.sinch.com/tutorials/android-messaging-tutorial-using-sinch-and-parse/, accedido: 12/01/2016.

[14] "TEST de ALCOHOLEMIA: cómo calcular la cantidad de alcohol," http://guardiasresis.blogspot.com.es/2014/07/test-de-alcoholemia-como-calcular-la.html, accedido: 17/01/2016.

[15] "Tasa de alcoholemia," https://investigayaprende.wordpress.com/fisica/tasa-de-alcoholemia/, accedido: 25/01/2016.

[16] D. M. M. Sánchez, D. B. C. Martín, P. J. R. H. González *et al.*, *Metodología de desarrollo de sistemas de información*.

[17] J. A. Gutiérrez de Mesa and C. Pagés Arévalo, *Planificación y gestión de proyectos informáticos*. Textos universitarios, Abril 2015.

[18] "Entorno de desarrollo Android (Android Studio)," http://www.sgoliver.net/blog/entorno-de-desarrollo-android-android-studio/, accedido: 15/03/2016.

[19] "Estructura de un proyecto Android (Android Studio)," http://www.sgoliver.net/blog/estructura-de-un-proyecto-android-android-studio/, accedido: 19/03/2016.

[20] "Componentes de una aplicación Android," http://www.sgoliver.net/blog/componentes-de-una-aplicacion-android/, accedido: 25/03/2016.

[21] "Desarrollando una aplicación Android sencilla (Android Studio)," http://www.sgoliver.net/blog/desarrollando-una-aplicacion-android-sencilla-android-studio/, accedido: 27/03/2016.

[22] A. L. Ayala, *Android Studio Curso Basico: Aprenda paso a paso*, 1st ed.

[23] N. Smyth, *Android Studio 2 Development Essentials*.

[24] A. Gerber and C. Craig, *Learn Android Studio: Build Android Apps Quickly and Effectively*. Apress.

[25] "Genymotion," https://www.genymotion.com/, accedido: 17/05/2016.

[26] S. Basulto, "Tutorial de Git en Español," blog.santiagobasulto.com.ar/programacion/2011/11/27/tutorial-de-git-en-espanol.html, accedido: 12/06/2016.

[27] "Android and iOS Squeeze the Competition, Swelling to 96.3 % of the Smartphone Operating System Market for Both 4Q14 and CY14, According to IDC," http://www.idc.com/getdoc.jsp?containerId=prUS25450615, Feb. 2015.

[28] P. P. Merino, "Android aumenta un 2 % su cuota de mercado en nuevos smartphones en España hasta un 90 %," http://ecommerce-news.es/actualidad/android-aumenta-2-cuota-mercado-nuevos-smartphones-espana-90-39857.html, Apr. 2016.

[29] "Anexo:Países por número de teléfonos móviles," https://es.wikipedia.org/wiki/Anexo:Pa%C3%ADses_por_n%C3%BAmero_de_tel%C3%A9fonos_m%C3%B3viles, accedido: 25/06/2015.

[30] "SQL," http://www.w3schools.com/sql/, accedido: 13/02/2016.

[31] A. Beaulieu, *Aprende SQL*, 2nd ed. Anaya.

[32] "HTML(5) Tutorial," http://www.w3schools.com/html/default.asp, accedido: 19/02/2016.

[33] A. Á. García, *HTML5 (Manuales imprescindibles)*. Anaya.

[34] "JavaScript Tutorial," http://www.w3schools.com/js/default.asp, accedido: 15/02/2016.

[35] A. d. C. Parra, *Javascript - Edición 2016 (Guías Prácticas)*. Anaya, 2016.

[36] "PHP 5 Tutorial," http://www.w3schools.com/php/default.asp, accedido: 23/02/2016.

[37] O. Heurtel, *PHP 5.6. Desarrollar Un Sitio Web Dinámico e Interactivo*. eni ediciones.

[38] "CSS Tutorial," http://www.w3schools.com/css/default.asp, accedido: 12/03/2016.

[39] C. Aubry, *CSS3. Domine Los Estándares Web Con Las Hojas De Estilo*. eni ediciones.

[40] "Preferencias en Android I: Shared Preferences," http://www.sgoliver.net/blog/preferencias-en-android-i-shared-preferences/, accedido: 15/05/2016.

[41] "Tareas en segundo plano en Android (I): Thread y AsyncTask," http://www.sgoliver.net/blog/tareas-en-segundo-plano-en-android-i-thread-y-asynctask/, accedido: 12/10/2016.

[42] "CODIGO EN JAVA PARA LA CONEXIÓN CON POSTGRESQL," http://introduccionpostgres.blogspot.com.es/2010/11/codigo-en-java-para-la-conexion-con.html, accedido: 29/11/2015.

[43] Copernic, "Android Generate QR code and Barcode using Zxing," http://stackoverflow.com/questions/22371626/android-generate-qr-code-and-barcode-using-zxing, accedido: 28/03/2016.

[44] "Integrar Barcode Scanner en nuestra aplicación Android," https://www.adictosaltrabajo.com/tutoriales/qr-reader-android/#03, accedido: 17/03/2016.

[45] "Barcode Scanner," https://play.google.com/store/apps/details?id=com.google.zxing.client.android&hl=es, accedido: 25/06/2015.

[46] "Dashboards," https://developer.android.com/about/dashboards/index.html?hl=es, accedido: 13/04/2016.

[47] "Notificaciones en Android (III): Diálogos," http://www.sgoliver.net/blog/notificaciones-en-android-iii-dialogos/, accedido: 18/04/2016.

[48] "Campañas 2007- Alcohol y menores. El alcohol te destroza por partida doble," http://www.msssi.gob.es/campannas/campanas07/alcoholmenores10.htm, accedido: 17/06/2016.

I

Glosario

Acrónimos

CSS	Cascading Style Sheets
HTML	HyperText Markup Language
PHP	Pre Hypertext Processor
QR	Quick Response
SQL	Structured Query Language
TPV	Terminal Punto de Venta

DEFINICIONES

Framework: Es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software.

Multicast: Consiste en el envío de la información en múltiples redes a múltiples destinos simultáneamente.

Protocolo de comunicación: Es un término utilizado en el mundo de la informática, para dar nombre a una serie de normas y criterios, los cuales, son utilizados para mantener una comunicación entre los dispositivos que forman parte de una red informática, es decir, entre los dispositivos que se encuentran conectados entre si por cualquier sistema de comunicación, sea alámbrico o inalámbrico.

Código QR: Es un módulo para almacenar información en una matriz de puntos o en un código de barras bidimensional.

II

Anexos

A

ANDROID

Android, a fecha de impresión de este documento, ha comercializado 16 versiones cuyo detalle se muestra en la tabla A.1. El uso que han tenido estas versiones se detalla en el gráfico de la figura A.1 [46].

Versión	Fecha de publicación	Nivel API	Nombre
Beta	05/11/2007		
1.0	23/09/2008	1	Astro
1.1	09/02/2009	2	Bender
1.5	30/04/2009	3	Cupcake
1.6	15/09/2009	4	Donut
2.0 / 2.1	26/10/2009	5(2.0) 6(2.0.1) 7(2.1.x)	Eclair
2.2	20/05/2009	8	Froyo
2.3	06/12/2010	9(2.3 - 2.3.2) 10(2.3.3 - 2.3.7)	Gingerbread
3.0 / 3.1 / 3.2	22/02/2011	11(3.0.x) 12(3.1) 13(3.2)	Honeycomb
4.0	19/10/2011	14(4.0 - 4.0.2) 15(4.0.3 - 4.0.4)	Ice Cream Sandwich
4.1	09/07/2012	16(4.1.1)	Jelly Bean
4.2	13/11/2012	17(4.2)	Jelly Bean
4.3	24/07/2013	18	Jelly Bean
4.4	31/11/2013	19	KitKat
5.0	12/11/2014	21	Lollipop
6.0	05/10/2015	23	Marshmallow

Tabla A.1: Versiones de Android, fechas de publicación y nombres.

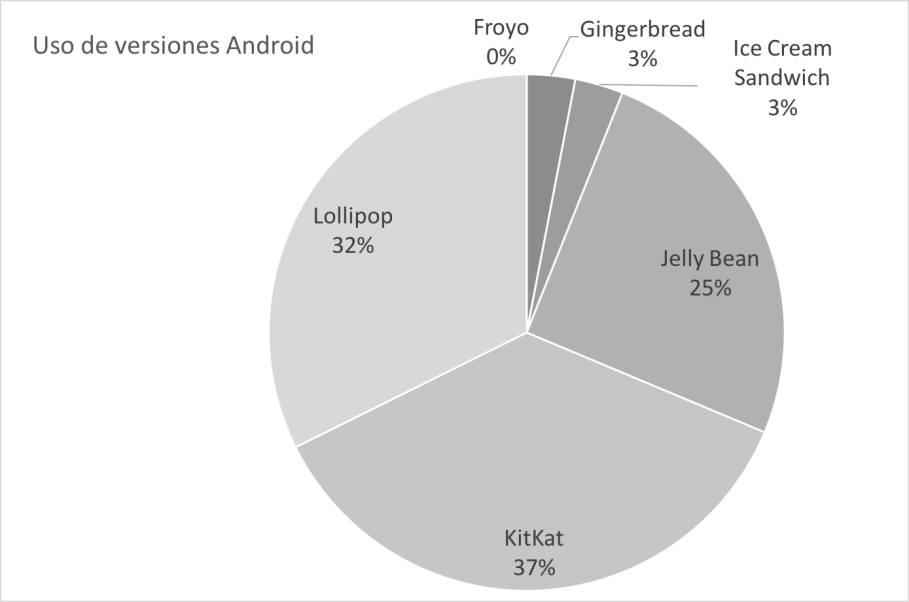


Figura A.1: Uso de versiones Android en Mayo de 2016.

B

Envío de mensajes

```
package com.example.hada.thewaiter.chat;

import com.example.hada.thewaiter.MainActivity;
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;

public class SendMulticast {

    public static void sendMulticast(String dir, int port, String msgout) throws
         IOException {

        byte[] buf = msgout.getBytes();
        InetAddress address = InetAddress.getByName(dir);
        DatagramSocket sendSocket = new DatagramSocket();
        DatagramPacket msgPacket = new DatagramPacket(buf, buf.length, address, port);
        sendSocket.send(msgPacket);
        sendSocket.close();
    }

    public static void sendMulticast(int port, String msgout) throws IOException {

        byte[] buf = msgout.getBytes();
        InetAddress address = InetAddress.getByName(MainActivity.getMiRed().getIp());
        DatagramSocket sendSocket = new DatagramSocket();
        DatagramPacket msgPacket = new DatagramPacket(buf, buf.length, address, port);
        sendSocket.send(msgPacket);
        sendSocket.close();
    }

    public static void sendMulticast(String msgout) throws IOException {

        byte[] buf = msgout.getBytes();
        InetAddress address =
             InetAddress.getByName(MainActivity.getMiRed().getMulticast());
        DatagramSocket sendSocket = new DatagramSocket();
        DatagramPacket msgPacket = new DatagramPacket(buf, buf.length, address,
             MainActivity.getMiRed().getPuerto());
        sendSocket.send(msgPacket);
        sendSocket.close();
    }
}
```

Código B.1: Código utilizado para el envío de mensajes a la red local.

C

Recepción de mensajes

```
package com.example.hada.thewaiter.chat;

import android.os.AsyncTask;
import android.util.Log;

import com.example.hada.thewaiter.MainActivity;
import com.example.hada.thewaiter.OnMensajeVacioListener;
import com.example.hada.thewaiter.OnRecepcionMensajePrivadoListener;
import com.example.hada.thewaiter.OnRespuestaPerfilListener;
import com.example.hada.thewaiter.OnSolicitudPerfilListener;
import com.example.hada.thewaiter.camarero.Notificacion;
import com.example.hada.thewaiter.camarero.NotificacionesActivity;
import com.example.hada.thewaiter.codigoQR.DatosQR;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.InetAddress;
import java.net.MulticastSocket;

// Para iniciar el thread new ReadMulticastRead().execute(Activity) dentro del onCreate de
    la
// actividad correspondiente.
//
// Solo es necesario implementar el metodo de la activity correspondiente que procesa el
    mensaje en
// onProgressUpdate que es el ultimo metodo que hay definido en esta clase. Si es necesario,
    como
// se pasa la activity, se puede usar activity.metodo sin necesidad de importar la clase.
//
// Para entender como funciona una AsyncTask:
// http://developer.android.com/reference/android/os/AsyncTask.html

public class AsyncMulticastRead extends AsyncTask<Void, String, Void> {

    private MulticastSocket receiveSocket;
    private static final String DEFAULT_IP = ``224.0.0.1'';
    private static final Integer DEFAULT_PORT = 5355;
    private static final Integer DEFAULT_TIMEOUT = 0; // Si es 0 no hay timeout

    private static OnRecepcionMensajePrivadoListener listener;
    private static OnSolicitudPerfilListener listener2;
    private static OnRespuestaPerfilListener listener3;
    private static OnMensajeVacioListener listener4;
```

```
private String receiveMulticast() {

    String recMsg;

    try {

        byte[] buf = new byte[10000];
        int i;

        // Define el datagrama donde se va a recibir
        DatagramPacket msgPacket = new DatagramPacket(buf, buf.length);

        // Se recibe del socket
        receiveSocket.receive(msgPacket);

        byte[] buf2 = new byte[msgPacket.getLength()];

        for (i = 0; i < msgPacket.getLength(); ++i)
            buf2[i] = buf[i];

        recMsg = new String(buf2);
        return recMsg;

    } catch (IOException e) {

        e.printStackTrace();
        return null;
    }
}

@Override
protected Void doInBackground(Void... params) {

    String recMsg;

    while (true) {

        recMsg = receiveMulticast();
        publishProgress(recMsg); // Fuerza la llamada a onProgressUpdate

        if (isCancelled()) break;

    }

    return null;
}

// Metodo que se ejecuta antes de lanzar el hilo
// protected void onPreExecute(Activity act) {

@Override
protected void onPreExecute() {

    InetAddress address;
```

```
        try {

            // Creacion de un socket multicast con timeout
            address = InetAddress.getByName(DEFAULT_IP);
            receiveSocket = new MulticastSocket(5355);

            if (DEFAULT_TIMEOUT != 0)
                receiveSocket.setSoTimeout(DEFAULT_TIMEOUT);
            receiveSocket.joinGroup(address);

        } catch (IOException e) {

            e.printStackTrace();

        }
    }

    // En esta situacion no puede ser llamado nunca
    // protected void onPostExecute() { receiveSocket.close(); }

    @Override
    protected void onCancelled(Void result) {
        receiveSocket.close();
    }

    protected void onProgressUpdate(String... recMsgs) {

        for (String s : recMsgs) {

            // Aqui se introduce la llamada al metodo de la UI que parsea y presenta,
            // la String receivedMsg es la String a parsear por el metodo y por tanto es
            // el parametro con el que se debe llamar al metodo.

            // Comprobamos el destinatario del mensaje
            String[] tokens = s.split("\\|");
            Log.i("TOKENS", tokens[0]);

            if (tokens[0].equals("GLOBALCHAT")) {

                ConversacionActivity.ImprimeMensaje(tokens[1]);

            } else if (tokens[0].equals("CAMARERO")) {

                String tokens2[] = tokens[1].split(":::");

                Notificacion n = null;
                Notificacion not = null;

                if(tokens2.length == 2){

                    n = new Notificacion(0, tokens2[0], Integer.parseInt(tokens2[1]));
                }

                int encontrado = 0;
```

```
            if (DatosQR.getMesa() == -1) { // Si el mensaje lo recibe el camarero

                if (MainActivity.getAdaptadorCamarero().getCount() != 0) {

                    for (int i = 0; i < MainActivity.getAdaptadorCamarero().getCount();
                            i++) {

                        not = MainActivity.getAdaptadorCamarero().getItem(i);

                        if(n.getTexto().equals(not.getTexto()) && n.getMesa() ==
                                not.getMesa()){

                            // Ya existe esta peticion
                            not.setPrioridad(not.getPrioridad() + 1);
                            encontrado = 1;
                            MainActivity.getAdaptadorCamarero().notifyDataSetChanged();

                        }

                    }

                    if (encontrado != 1)
                        MainActivity.getAdaptadorCamarero().add(new Notificacion(1,
                                n.getTexto(), n.getMesa()));

                } else

                    MainActivity.getAdaptadorCamarero().add(new Notificacion(1,
                            n.getTexto(),
                                n.getMesa()));

                // Actualizamos la vista de las notificaciones para el camarero
                NotificacionesActivity.ImprimeNotificaciones();

            }

        } else if (tokens[0].equals("PERFIL")) {

            String tokens2[] = tokens[1].split(":::");

            listener2.onSolicitudPerfil(tokens2[0], tokens2[1]);

        } else if (tokens[0].equals("PRIVADO")){

            String tokens2[] = tokens[1].split(":::");

            // Si estoy conectada envio mi apodo
            if(tokens2.length == 3)
                listener.onRecepcionMensajePrivado(tokens2[0], tokens2[1], tokens2[2]);

            else
                listener4.onMensajeVacio(tokens2[0]);

        } else if(tokens[0].equals("RESPUESTA_PERFIL")) {

            String tokens2[] = tokens[1].split(":::");

            listener3.onRespuestaPerfil(tokens2[0], tokens2[1], tokens2[2], tokens2[3],
                    tokens2[4]);

        }
```

```
                else
                    return;

            }

        }

        public static void setOnRecepcionMensajePrivado(OnRecepcionMensajePrivadoListener
             l) {
            listener = l;
        }

        public static void setOnSolicitudPerfil(OnSolicitudPerfilListener l) {
            listener2 = l;
        }

        public static void setOnRespuestaPerfil(OnRespuestaPerfilListener l) {
            listener3 = l;
        }

        public static void setOnMensajeVacio(OnMensajeVacioListener l) {
            listener4 = l;
        }

    }
```

Código C.1: Código utilizado para la recepción y el filtrado de mensajes que han sido lanzados a la red local.

D

Página web

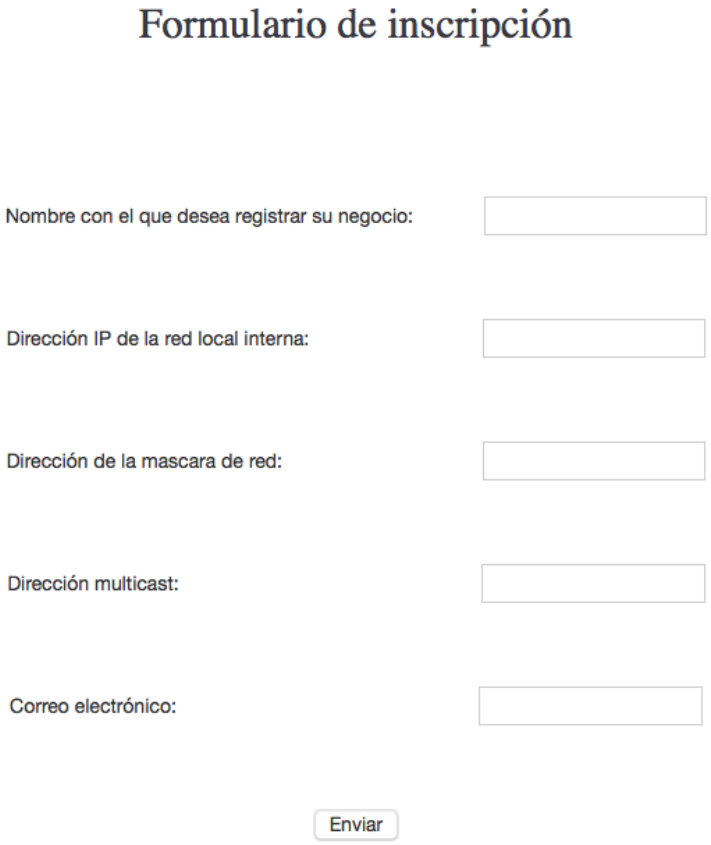


Figura D.1: El formulario de registro aparece como primera pantalla de la página web y permite que los usuarios den de alta sus negocios al introducir de forma correcta la información solicitada.

Figura D.2: La confirmación del registro aparece como segunda pantalla de la página web y ofrece información para que el usuario pueda acceder y configurar los datos de su negocio desde la aplicación.

E

Conexión con la base de datos

```
import java.sql.Connection;
import java.sql.DriverManager;

public class Conectar {

    private static Connection conexion = null;

    public static int establecerConexion(String usuario, String password, String url) {

        if (conexion != null) // Conexion establecida
            return 0;

        try {

            Class.forName("org.postgresql.Driver");
            conexion = DriverManager.getConnection(url, usuario, password);

            if (conexion != null)
                return 1;

        } catch (Exception e) {

            LogSistema.infoLog(e.getMessage());
            return -1;
        }

        return 0;

    }

    public static Connection getConexion() {
        return conexion;
    }

    public void setConexion(Connection conexion) {
        this.conexion = conexion;
    }

}
```

Código E.1: Código utilizado para la gestión de la conexión con la base de datos

F

Inserción de imágenes en la base de datos

```
public static void main(String[] args) {

    String usuario = "thewaiter";
    String password = "7fgL0r3n4";
    String url = "jdbc:postgresql://metis.ii.uam.es:5432/thewaiter";
    Connection conexion = null;

    int comprobanteConexion = Conectar.establecerConexion(usuario, password, url);

    if (comprobanteConexion == 0 || comprobanteConexion == 1) {

        // Establecimiento de la conexion
        conexion = Conectar.getConexion();

        insertarImagen(40, conexion);


    } else {

        System.out.println("La conexion con la base de datos ha fallado.");

    }

}
```

```
public static void insertarImagen(int numero, Connection conexion) {

    try {

        File file = new File("src/imagenes/foto_" + numero + ".jpg");
        FileInputStream fis = new FileInputStream(file);
        PreparedStatement ps = conexion.prepareStatement("UPDATE productos_1 SET
            foto=? where id=?");

        ps.setBinaryStream(1, fis, file.length());
        ps.setInt(2, numero);
        ps.execute();
        ps.close();
        fis.close();

        System.out.println("Inserteee!!");

    } catch (IOException e) {
        System.out.println(e.getMessage());
    } catch (SQLException e) {
        System.out.println(e.getMessage());
    }

}
```

Código F.1: Código utilizado para la inserción de imágenes en la base de datos

G

Generación de códigos QR

```
private void generateQRCode_general(String data, ImageView img)throws WriterException
    {

    com.google.zxing.MultiFormatWriter writer = new com.google.zxing.MultiFormatWriter();
    String finaldata = Uri.encode(data, "utf-8");

    BitMatrix bm = writer.encode(finaldata, BarcodeFormat.QR_CODE, 150, 150);
    Bitmap ImageBitmap = Bitmap.createBitmap(150, 150, Bitmap.Config.ARGB_8888);

    for (int i = 0; i < 150; i++) { // Width

        for (int j = 0; j < 150; j++) { // Height

        ImageBitmap.setPixel(i, j, bm.get(i, j) ? Color.BLACK: Color.WHITE);

        }

    }

    if (ImageBitmap != null) {

        img.setImageBitmap(ImageBitmap);

    } else {

        Toast.makeText(getApplicationContext(), "Error al generar el codigo QR",
            Toast.LENGTH_SHORT).show();

    }

}
```

Código G.1: Código utilizado para generar los códigos QR de la aplicación

H

Flujo de peticiones de los clientes a los camareros

H.0.4. Introducción

En este apartado se detalla una de las funcionalidades del protocolo de comunicación diseñado que consiste en permitir que lo usuarios realicen peticiones desde su dispositivo a los empleados del establecimiento.

H.0.5. Flujo de las peticiones

Figura H.1: En este diagrama se muestra el flujo de las peticiones que los clientes pueden enviar a los empleados de un establecimiento.

El detalle del flujo puede observarse en la figura H.1 y a continuación se expondrá paso a paso para facilitar el entendimiento del diagrama:

1.– Camarero. En esta pantalla se concentran las solicitudes más habituales: atención, limpieza

de la zona, petición personalizada (por ejemplo, para pedir un salero).

2.– Pedido. En esta pantalla se puede navegar por la carta añadiendo productos al pedido, acceder al pedido actual y al pedido personalizado.

3.– Mi pedido. En esta pantalla se listan los productos que el usuario haya ido añadiendo y podrá enviarselo al camarero. Las cantidades de los productos añadidos son editables.

4.– Pedido especial. En esta pantalla el usuario puede escribir un texto libre con un pedido por si requiere una modificación de los productos de la carta.

5.– Pago. En esta pantalla se permite que un usuario solicete la cuenta, especificando si pagaŕa en efectivo o de forma electrónica.

6.– Notificaciones. En esta pantalla el camarero recibe todas las notificaciones de los clientes. Cada notificación contiene detalles sobre lo que el cliente solicita, la mesa desde la cual se solicita y la prioridad.

Flujo de mensajes privados

I.0.6. Introducción

En este apartado se detalla una de las funcionalidades del protocolo de comunicación diseñado que consiste en permitir que lo usuarios se comuniquen de forma privada gracias a un hilo de mensajes que pueden abrir y cerrar cuando lo deseen.

Estos mensajes no aparecen en contexto ni se almacenan, el objetivo era proporcionar una conversación completamente privada, tanto que los propios usuarios no tendrán acceso a más mensajes que el actual que estén enviando o hayan recibido.

I.0.7. Flujo de los mensajes

El detalle del flujo puede observarse en la figura I.1 y a continuación se expondrá paso a paso para facilitar el entendimiento del diagrama:

1.– En esta pantalla se visualiza el chat de Alice que quiere iniciar una mensajería privada con Charly que también está escribiendo en el chat global por lo que pulsa sobre cualquiera de sus mensajes.

2.– Alice es notificada de que va a solicitarle información sobre el perfil a Charly, lo que les permitirá iniciar un hilo de mensajes privados. Acepta el dialogo [47] y se envía la petición.

3.– Charly acepta enviarle a Alice la información de su perfil a través del diálogo que se ha abierto en su pantalla al recibir la solicitud de Alice.

4.– Alice recibe el perfil de Charly y puede comenzar a enviar mensajes privados.

5.– Charly recibe un mensaje privado de Alice y responde.

6.– Alice recibe un mensaje privado de Charly.

Los pasos 6 y 7 se pueden repetir tantas veces como los usuarios deseen hasta que en uno de los mensajes recibidos en lugar de responder pulsen cerrar, con lo que cerrarán el hilo de comunicación y si quieren volver a comunicarse tendrán que repetir el proceso desde el paso 1.

Figura I.1: Diagrama del flujo de los mensajes privados que pueden lanzarse entre dos usuarios.

J

Encuesta para los usuarios

Encuesta sobre la eficiencia actual de los negocios dedicados a la hostelería.

Estos datos se utilizarán para estimar la eficiencia actual de los negocios de hostelería de forma que pueda comparar estos datos con los obtenidos implantando en dichos negocios el proyecto que elaboraré como mi Trabajo de Fin de Grado.

Indique su D.N.I para comprobar que no se han falseado los datos de la encuesta por favor:

.....

Aviso legal sobre la protección de datos: estos datos serán utilizados por Lorena Aguilar Briz para realizar una serie de gráficos que se expondrán en la memoria del proyecto. De ser necesaria la comprobación de estos datos podrían estar expuestos a los componentes del tribunal.

1. Indique el numero de VISITAS que realizó los 7 días previos a la realización de la encuesta a negocios de hostelería.
Cafeterías, restaurantes, establecimientos de comida rápida, etc.

.....

¡Importante! Todas las visitas cuentan, aunque sean al mismo establecimiento.

2. Tiempo medio que tardó el servicio del establecimiento en dejar su mesa o su zona limpia de consumiciones anteriores o desperdicios.

.....

3. Tiempo medio en el que el servicio del establecimiento inició el primer contacto con usted, para anotar su petición.

.....

Figura J.1: Encuesta realizada a los usuarios para conocer sus opiniones sobre determinados aspectos de los negocios de hostelería en la actualidad (parte 1).

4. Tiempo medio hasta que el servicio le entregó el pedido completo.

.....

5. Tiempo medio que esperó para pedir la cuenta más el tiempo que después de pedida tardo en recibirla más el tiempo que esperó hasta recibir su cambio si fuese necesario.

.....

6. Numero medio de errores que el servicio cometió con el pedido que usted hizo.

.....

7. ¿Cree que unas fechas en las que estos negocios se encuentren especialmente concurridos (saturados) de clientes han podido afectar a los datos de su encuesta?

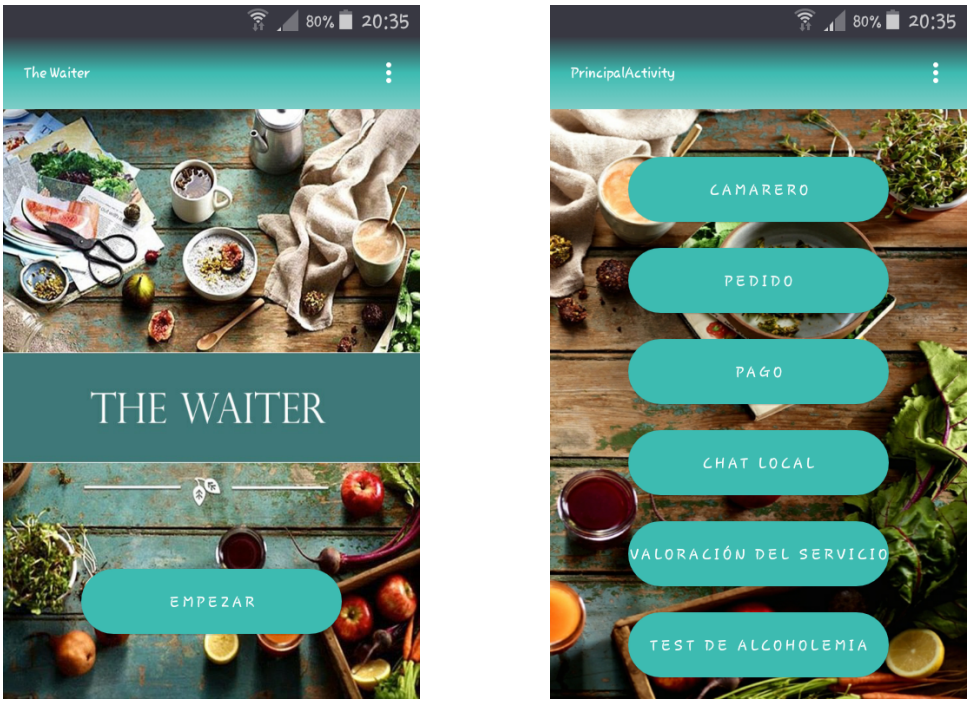
.....

¡Gracias por participar!

Figura J.2: Encuesta realizada a los usuarios para conocer sus opiniones sobre determinados aspectos de los negocios de hostelería en la actualidad (parte 2).

K

Imágenes de la aplicación



(a) Pantalla principal

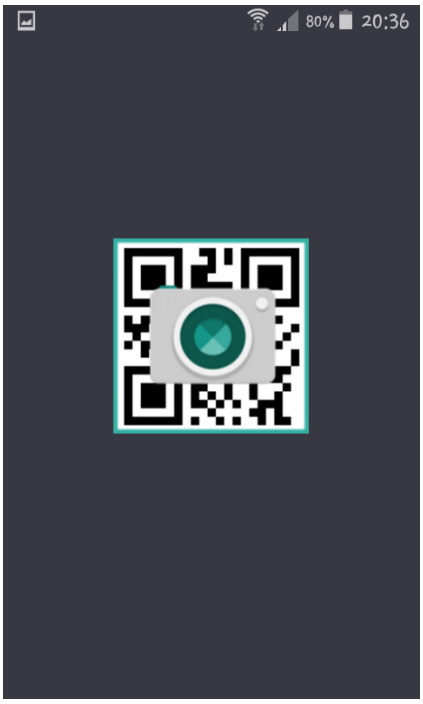
(b) Menú principal

Figura K.1: Imágenes de la aplicación (parte 1): Pantalla principal y menú principal

(a) Configuración de perfil

(b) Configuración de perfil

Figura K.2: Imágenes de la aplicación (parte 2): Configuración de perfil

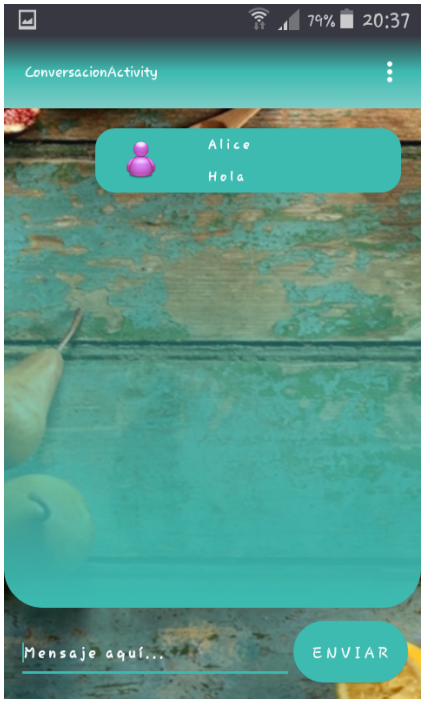


(a) Escaner de QR

(b) Escaner de QR

Figura K.3: Imágenes de la aplicación (parte 3): Escaner de QR

(a) Chat global



(b) Chat global

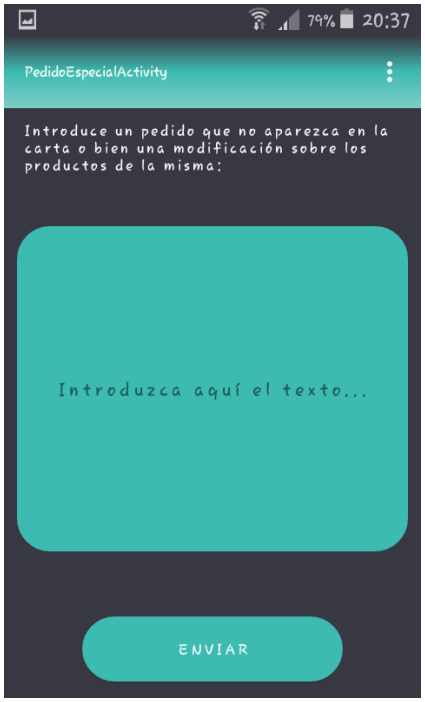
Figura K.4: Imágenes de la aplicación (parte 4): Chat global

(a) Menú de camarero

(b) Menú de pago

Figura K.5: Imágenes de la aplicación (parte 5): Menú de camarero y menú de pago

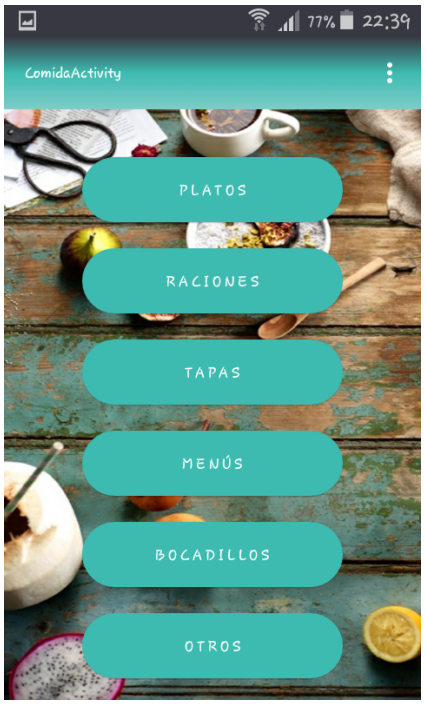
(a) Menú de pedido



(b) Formulario de pedido especial

Figura K.6: Imágenes de la aplicación (parte 6): Menú de pedido y formulario de pedido especial

(a) Menú de bebidas



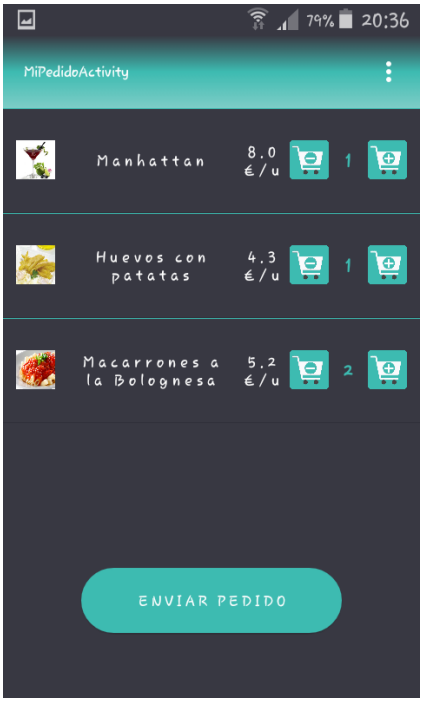
(b) Menú de comida

Figura K.7: Imágenes de la aplicación (parte 7): Menú de bebidas y menú de comida

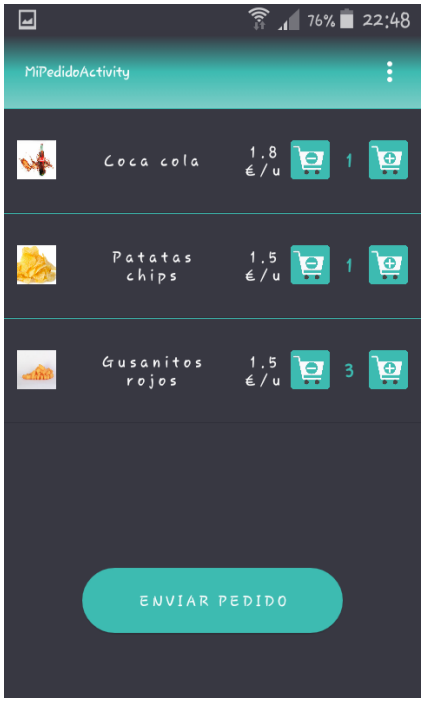
(a) Carrito

(b) Detalle del producto

Figura K.8: Imágenes de la aplicación (parte 8): Carrito y detalle del producto

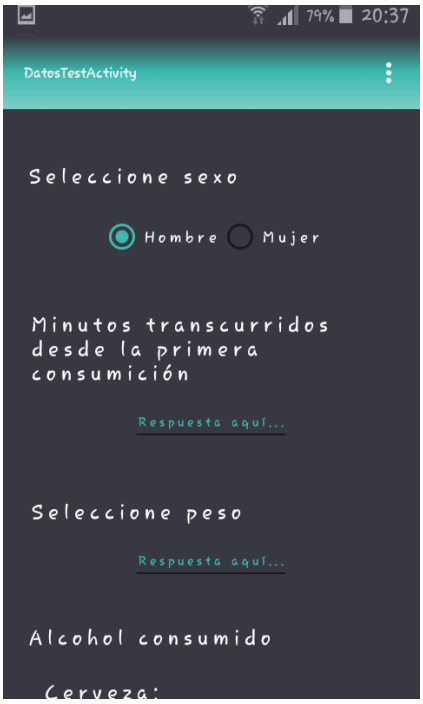


(a) Mi pedido



(b) Mi pedido

Figura K.9: Imágenes de la aplicación (parte 9): Mi pedido y modificación de mi pedido



(a) Test de alcoholemia

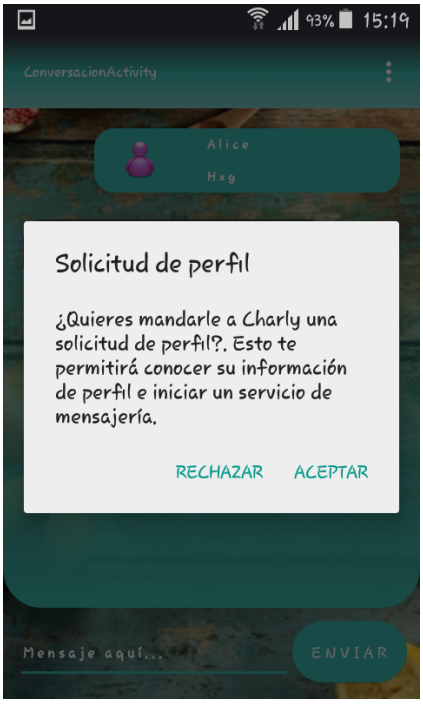
(b) Test de alcoholemia

Figura K.10: Imágenes de la aplicación (parte 10): Formulario del test de alcoholemia

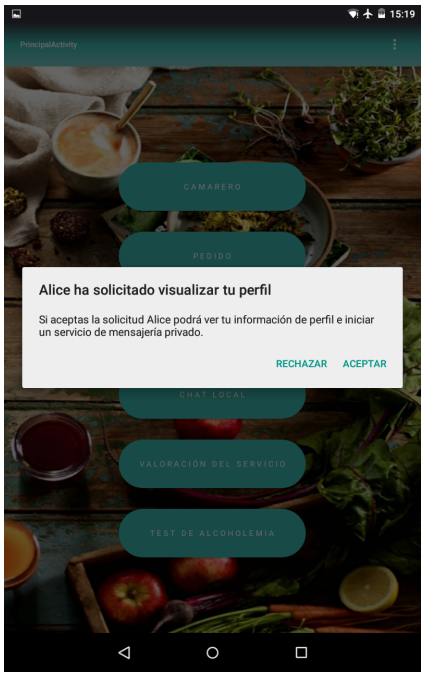
(a) Resultado del test de alcoholemia

(b) Resultado del test de alcoholemia

Figura K.11: Imágenes de la aplicación (parte 11): Resultado del test de alcoholemia



(a) Solicitud de perfil



(b) Recepción de la solicitud de perfil

Figura K.12: Imágenes de la aplicación (parte 12): Solicitud del perfil

(a) Recepción del perfil

(b) Mensajería instantánea

Figura K.13: Imágenes de la aplicación (parte 13): Recepción del perfil y mensajería instantánea

L

Test de alcoholemia

L.0.8. Introducción

El test de alcoholemia es uno de los elementos del proyecto. En su origen se implementó para garantizar, en la medida de lo posible, la seguridad de los usuarios que ingieran bebidas alcoholicas en los establecimientos de hosteleria. Tras las pruebas realizadas se ha detectado que es utilizada por los usuarios con fines recreativos. Aunque no se puede estivar la tasa de alcoholemia de forma exacta, como lo harían unos análisis de sangre, las fórmulas utilizadas para este cálculo son muy efectivas, siempre que los datos se acerquen lo más posible a la realidad. Las fuentes han sido obtenidas de la página web oficial del Ministerio de Sanidad, Servicios Sociales e Igualdad [48].

L.0.9. Datos

El cálculo de la tasa de alcoholemia de los usuarios se ha realizado basándonos en los datos que se muestran a continuación:

Número de UBE	Alcoholemia Grs./l	Efectos fisicos y psicologicos
1	0,2 - 0,3	Sin efectos evidentes. Ligera elevación del estado de ánimo.
2	0,5 - 0,6	Relajación, calor, disminución del tiempo de reacción y de la coordinación fina.
5	1,4 - 1,5	Alteración mayor del control físico y mental: habla y visión difíciles.
7	2	Pérdida del control motor (requieren de ayuda) confusión mental.
10	3	Intoxicación severa; control conciente mínimo.
14	4	Inconsciencia; umbral del estado de coma.
17	5	Coma profundo.
20	6	Muerte por depresión respiratoria.

Tabla L.1: Detalle de los datos utilizados para el cálculo de la tasa de alcoholemia.

L.0.10. Cómo calcular la tasa de alcoholemia

Como las bebidas tienen distintas concentraciones de alcohol, las consumiciones se traducen a Unidades de Bebida Estándar (UBE): Una UBE equivale a 10 Grs. de alcohol.

Calculo de los gramos de alcohol:

Gramos de alcohol = Volumen (expresado en c.c.) x Graduacion x densidad / 100

- graduacion = Graduacion de etanol en la bebida
- densidad = Densidad de alcohol (0,8)

Una vez realizada esta operación aun no hemos obtenido la tasa de alcoholemia de un individuo, debemos aplicar factores referentes al peso y al sexo de dicho individuo de la siguiente forma:

El peso lo multiplicaremos por un factor de la siguiente forma:

- x 0,7 en el caso de los hombres
- x 0,6 en el caso de las mujeres

Para continuar se divide el resultado de la primera operacion entre el peso del individuo por el factor correspondiente a su sexo.

Finalmente, teniendo en cuenta que el alcohol no se almacena de forma indefinida en el cuerpo, se debe tener en cuenta el factor de eliminacion. Este factor es 0,0025 que tendrá que ser multiplicado por el número de minutos transcurridos desde la primera consumición.

El dato final se obtiene restándole al dato calculado previamente en la operación anterior de la tasa se eliminacion (minutos transcurridos desde la primera consumición x 0,0025).

Ahora se tiene la tasa en Gramos/Litro y se aplica la tabla de medidas que se prporcionó al principio del apartado para estimar los resultados de la tasa del alcoholemia sobre un determinado individuo.

Es importante recordar que la tasa de alcoholemia permitida en sangre para conductores está por debajo de 0,5 g/L en sangre (o lo que es lo mismo, 0,25 mg/L en aire espirado), y para noveles y profesionales por debajo de 0,3 g/L.